

**Laslo Kraus
Igor Tartalja**

Zbirka zadataka

IZ

**Projektovanja
softvera**

AKADEMSKA MISAO

Beograd, 2013

Laslo Kraus
Igor Tartalja

ZBIRKA ZADATAKA
IZ PROJEKTOVANJA SOFTVERA
Treće, dopunjeno izdanje

Recenzenti
Dr Milo Tomašević
Dr Vladimir Blagojević

Izdavač
AKADEMSKA MISAO
Bulevar kralja Aleksandra 73, Beograd

Lektor
Anđelka Kovačević

Dizajn korica
Zorica Marković, akademski slikar

Štampa
Akademska misao, Beograd

Tiraž
200 primeraka

ISBN 978-86-7466-475-9

NAPOMENA: Fotokopiranje ili umnožavanje na bilo koji način ili ponovno objavljivanje ove knjige - u celini ili u delovima - nije dozvoljeno bez prethodne izričite saglasnosti i pismenog odobrenja izdavača.

Predgovor

Ova zbirka sadrži sve zadatke koji su u toku školske 2012/13. godine rađeni na vežbama iz predmeta **Projektovanje softvera** na četvrtoj godini *Odseka za računarsku tehniku i informatiku*, odnosno na trećoj godini *Odseka za softversko inženjerstvo Elektrotehničkog fakulteta Univerziteta u Beogradu*. Pored toga, uključeni su zadaci i pitanja sa većine kolokvijuma i ispita. Iako je prvenstveno namenjena studentima *Elektrotehničkog fakulteta*, koji prate predmet **Projektovanje softvera**, zbirka može biti od koristi i drugima koji uče da projektuju softver modelovanjem na jeziku UML, uz primenu projektnih uzoraka.

U odnosu na prvo i drugo izdanje zbirke dodati su zadaci sa kolokvijuma *Odseka za računarsku tehniku i informatiku* {K}, prvog i drugog kolokvijuma *Odseka za softversko inženjerstvo* {K1, K2} i ispita oba odseka {I}. Nekoliko ispitnih zadataka je nastalo na osnovu ideja kolega Žarka Stanisavljevića {Ž.S.}, Nemanje Kojića {N.K.}, Živojina Šuštrana {Ž.Š.} i Đorda Đurđevića {Đ.Đ.}.

Prvi deo zbirke sadrži rešene zadatke, organizovane na način da prate auditorske vežbe iz predmeta, i većina njih se radi na vežbama u toku semestra. Svrha ovog dela zbirke jeste da studentima olakša praćenje nastave. Zadaci su priloženi u zatečenom obliku u kojem se koriste u toku izvođenja vežbi. To znači da je uz pojedina rešenja priloženo vrlo malo dopunskih objašnjenja i to prvenstveno u obliku slika ili formula. Zbog toga se studentima izričito preporučuje redovno pohađanje vežbi u toku nastave, gde će dobiti usmena objašnjenja za lakše razumevanje pojedinih zadataka.

U drugom delu zbirke nalaze se ispitni zadaci bez rešenja. Namenjeni su za samostalno rešavanje u toku priprema za ispite.

U trećem delu zbirke nalaze se pitanja za proveru znanja sa ispita i kolokvijuma iz predmeta. Pitanja su uređena tematski, ali se napominje da ona ne čine skupove pitanja koji potpuno pokrivaju odgovarajuće teme. Odgovori na pitanja koji se očekuju od studenata na kolokvijumima i ispitima treba da budu jezgroviti i precizni. Primer odgovora na nekoliko pitanja, priložen je na kraju ovog dela.

Iako su problemi koncipirani i formulisani na način da rešenja ne zavise od konkretnog alata za projektovanje softvera, za njihovo rešavanje korišćen je alat za modelovanje *StarUML* 5.0.2, koji može da se preuzme sa adrese <http://staruml.sourceforge.net/> i besplatno koristi. Skreće se pažnja na to da *StarUML* nije potpuno kompatibilan sa jezikom UML verzije 2, kao i da ima programskih grešaka (naročito u generisanju koda), ali je u kategoriji besplatnih alata jedan od najudobnijih za korišćenje i u prihvatljivoj meri poštuje standard *UML* 2.

Svoja zapažanja čitaoci mogu da upute autorima elektronskom poštom na adrese kraus@etf.rs i tartalja@etf.rs.

Beograd, septembar 2013.

Laslo Kraus
i Igor Tartalja

Sadržaj

Predgovor	3
Sadržaj	4
Preporučena literatura	6
Rešeni zadaci	7
Zadatak 1 Proizvođač, potrošač i skladište predmeta (osnove dijagrama <u>klasa</u>)	8
Zadatak 2 Tačka i figure u ravni (nasledivanje i apstraktne klase)	12
Zadatak 3 Uređivači uporedivih stvari (interfejsi)	16
Zadatak 4 Proizvod, skladište, mehanizmi i motor (dijagram <u>paketa</u> ; aktivna klasa)	20
Zadatak 5 Samoposluga (generisanje modela na osnovu programa na jeziku <i>Java</i>)	24
Zadatak 6 Proizvodi, mašine i radnik (generisanje modela na osnovu programa na jeziku <i>C++</i>)	28
Zadatak 7 Pravila, student, studentski odsek (projektni uzorak <i>Unikat</i>)	30
Zadatak 8 Jednostavni predmeti, sklopovi i radnici (projektni uzorci <i>Prototip</i> i <i>Sastav</i>)	32
Zadatak 9 Jednostavni predmeti, sklopovi i radnici (dijagrami <u>objekata</u> i <u>interakcije</u>)	34
Zadatak 10 Otpornici {K1, 10.11.2006.}	36
Zadatak 11 Proizvod, skladište, mehanizmi i motor	38
Zadatak 12 Vektor, predmeti, orijentisani predmeti, struktura i radnik {K1, 01.11.2007.}	40
Zadatak 13 Simbol, font, vektor, figure, crtež, platno i aplikacija {K1, 30.10.2008.}	44
Zadatak 14 Tačka, figure, krive, familija krivih i crtež {K1, 30.10.2009.}	48
Zadatak 15 Potrošači, elektrane, elektrodistribucija {K1, 1.11.2010.}	52
Zadatak 16 Lavirint {K1, 28.10.2011.}	56
Zadatak 17 Lica, naučni radovi, naučne oblasti i zbornici radova {K1, 23.10.2012.}	60
Zadatak 18 Samoposluga	64
Zadatak 19 Distributeri, klijenti, pošiljke i raspodele (projektni uzorak <i>Posmatrač</i>)	70
Zadatak 20 Boja, tačka, figure i obilasci (projektni uzorak <i>Iterator</i>)	78
Zadatak 21 Grafički statistički pokazatelji (projektni uzorak <i>Dopuna</i>)	88
Zadatak 22 Prodavnica, fakultet i pošta (dijagram <u>slučajeva korišćenja</u>)	92
Zadatak 23 Projektni uzorci <i>Strategija</i> i <i>Šablonska metoda</i>	94
Zadatak 24 Atomi, znak, piksel, tekst, slika i dokument {K, 01.12.2006.}	98
Zadatak 25 Artikli, zbirke artikala, obilasci, osobe, načini plaćanja, samoposluga i kamion {K, 16.11.2008.}	102
Zadatak 26 Polje, tabla, figure, igrači i igre {K, 30.11.2007.}	106
Zadatak 27 Uporediv, životinje, izbori, štala, obilasci štale, poslovi i osobe {K, 21.11.2009.}	110
Zadatak 28 Članci, osobe, e-novine {K, 27.11.2010.}	114
Zadatak 29 Berza poslova i radnici {K, 19.11.2011.}	118
Zadatak 30 Geometrijske figure, geografski elementi i karte {K, 18.11.2012.}	122
Zadatak 31 Samoposluga (dijagrami <u>aktivnosti</u> i <u>stanja</u>)	126
Zadatak 32 Predmeti i akteri	128
Zadatak 33 Predmet, skladište, pokazivač predmeta i nadzornici {K2, 15.12.2006.}	134

Zadatak 34	Roba, artikal, porez, popust, skladište, redosledi i izveštaj {K2, 06.12.2007.}	136
Zadatak 35	Vozila, mesto, parking, redosledi i semafor {K2, 04.12.2008.}	140
Zadatak 36	Biblioteka, publikacije, dela i osobe {K2, 04.12.2009.}	144
Zadatak 37	Dela, galerija, osobe i casovnik {K2, 07.12.2010.}	146
Zadatak 38	Akvizicija podataka {K2, 29.11.2011.}	150
Zadatak 39	Distribucija električne energije {K2, 27.11.2012.}	154
Zadatak 40	Dijagram stanja i aktivnosti niti u jeziku Java	158
Zadatak 41	Projektni uzorci u paketu AWT	159
Zadatak 42	Gradski saobraćaj (projektni uzorak <i>Stanje</i>)	160
Zadatak 43	Figure u ravni (projektni uzorak <i>Muva</i>)	166
Zadatak 44	Predmeti, mašine i radnik (projektni uzorak <i>Proizvodna metoda</i>)	172
Zadatak 45	Igre na tabli (projektni uzorci <i>Podsetnik</i> i <i>Komanda</i>)	176
Zadatak 46	Biblioteka (dijagram <u>komponenata</u> , projektni uzorci <i>Most</i> i <i>Apstraktna fabrika</i>)	184
Zadatak 47	Uredaj, operacije i API (projektni uzorak <i>Fasada</i>) {I, 14.02.2007.}	192
Zadatak 48	Datoteke, sistem datoteka i medijumi (projektni uzorak <i>Zastupnik</i>) {I, 20.01.2010.}	200
Zadatak 49	Osoba, zaposleni, firma, roman, autor, izdavačka kuća, kurir, izvršioci i nalog (projektni uzorci <i>Posrednik</i> i <i>Lanac odgovornosti</i>) {I, 10.01.2011.}	204
Zadatak 50	Preduzeće, radnici, skladište i automobili (projektni uzorak <i>Graditelj</i>) {I, 06.02.2009.}	210
Zadatak 51	Tačka, boja, grafički elementi, crtači, figure, komponente, akteri i program {I, 05.02.2007.}	216
Zadatak 52	Vozila, saobraćajnice i semafor {I, 23.01.2008.}	222
Zadatak 53	Tačka, boja, duži, telo, scena, crtači, preslikavanje, dugme, radnja i program {I, 28.01.2009.}	228
Zadatak 54	Berza rada {I, 17.01.2012. – Ž.Š.}	232
Zadatak 55	Merenje pulsa i pritiska {I, 15.01.2013. – Ž.Š.}	236

Ispitni zadaci241

Zadatak I	Izdavaštvo {I, 21.02.2007.}	242
Zadatak II	Videoteka {I, 08.03.2007.}	243
Zadatak III	Medicinski dokumenti {I, 04.07.2007.}	244
Zadatak IV	Testovi {I, 05.09.2007.}	245
Zadatak V	Šalterska služba {I, 29.02.2008.}	246
Zadatak VI	Softverska kompanija {I, 09.07.2008 – Ž.S.}	247
Zadatak VII	Upravljanje nitima i procesima {I, 03.09.2008.}	248
Zadatak VIII	Igre na tabli, šah {I, 17.09.2008.}	249
Zadatak IX	Prodaja računara {I, 09.07.2008. – Ž.S.}	250
Zadatak X	Čoveče ne ljuti se {I, 15.06.2009.}	251
Zadatak XI	Komunikacioni sistem {I, 12.02.2010., D.D.}	252
Zadatak XII	Sistem komunikacija za rezervisanje parkinga {I, 31.01.2011., N.K.}	253
Zadatak XIII	Veb forum {I, 27.08.2011. – N.K.}	254
Zadatak XIV	Dohvatanje internet stranica {I, 07.02.2012., Ž.Š.}	255
Zadatak XV	Elektronsko skladište {I, 05.02.2013. – Ž.Š.}	256

Ispitna pitanja257

i	UML – uvod	258
ii	UML – dijagrami klasa	258
iii	UML – dijagrami paketa	259
iv	UML – dijagrami objekata	259
v	UML – dijagrami interakcije	259
vi	UML – dijagrami slučajeva korišćenja	259
vii	UML – dijagrami aktivnosti	260

viii	UML – dijagrami stanja	260
ix	UML – dijagrami klasa, napredniji pojmovi	260
x	UML – dijagrami složene strukture	261
xi	UML – dijagrami komponenata	261
xii	UML – dijagrami raspoređivanja	261
xiii	UML – dijagrami pregleda interakcije	261
xiv	Projektni uzorci – Uvod	262
xv	Projektni uzorci stvaranja	262
xvi	Projektni uzorci strukture	262
xvii	Projektni uzorci ponašanja	263
xviii	Primeri odgovora na pitanja	264

Preporučena literatura

- [1] Grady Booch, James Rumbaugh, Ivar Jacobson: **The Unified Modeling Language User Guide**, 2nd Edition, Addison Wesley, 2005. (Prevod na srpski jezik: Booch.G., Rumbaugh.J., Jacobson.I., UML Vodič za korisnike, *prevod prvog izdanja*, CET, Beograd, 2001. – zastareo; obrađuje UML 1)
- [2] James Rumbaugh, Ivar Jacobson, Grady Booch, **Unified Modeling Language Reference Manual**, 2nd Edition, Addison-Wesley, 2004.
- [3] **Unified Modeling Language: Infrastructure i Unified Modeling Language: Superstructure** – Object Management Group, 2011., <http://www.omg.org/spec/UML/2.4.1/>
- [4] Erich Gamma, Richard Helm, Ralph Johnson, John Vlissides: **Design Patterns: Elements of Reusable Object-Oriented Software**, Addison-Wesley Professional Computing Series, Addison-Wesley Publishing, Inc., Reading, Massachusetts, 1995. (Prevod na srpski jezik: **Gotova rešenja: Elementi objektno orijentisanog softvera**, CET, Beograd, 2002.)
- [5] Alan Shalloway, James R. Trott: **Design Patterns Explained: A New Perspective on Object-Oriented Design**, 1st Edition, Addison Wesley Professional, 2002. (Prevod na srpski jezik: **Projektni obrasci – Nove tehnike objektno orijentisanog projektovanja**, Mikro knjiga, Beograd, 2004.)
- [6] Laslo Kraus: **Rešeni zadaci iz programskog jezika Java**, treće izdanje, Akademska misao, Beograd, 2012.
- [7] Laslo Kraus: **Rešeni zadaci iz programskog jezika C++**, treće izdanje, Akademska misao, Beograd, 2011.
- [8] Igor Tartalja: **Projektovanja softvera - skripta**, elektronsko izdanje, 2011, [http://rti.etf.bg.ac.rs/rti/ir4ps/predavanja/Projektovanja softvera - skripta.pdf](http://rti.etf.bg.ac.rs/rti/ir4ps/predavanja/Projektovanje%20softvera%20-%20skripta.pdf)

Rešeni zadaci

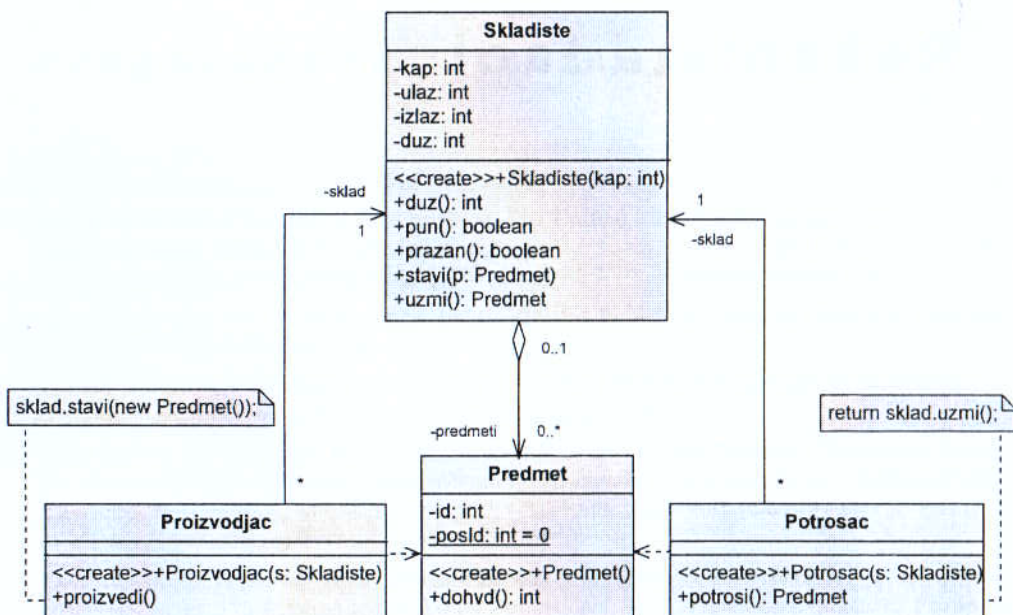
Zadatak 1 Proizvođač, potrošač i skladište predmeta (osnove dijagrama klasa)

Nacrtati dijagram klasa na jeziku *UML* sledećeg sistema klasa:

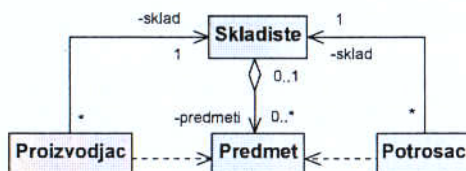
- **Predmet** ima jedinstven, automatski generisan celobrojan identifikator koji može da se dohvatiti.
- **Skladište** može da sadrži zadat broj predmeta. Stvara se prazno, posle čega predmeti mogu da se stavljaju i uzimaju jedan po jedan. Predmeti se uzimaju po redosledu stavljanja. Može da se dohvati broj predmeta u skladištu i da se ispita da li je skladište puno i da li je prazno.
- **Proizvođač** može da napravi jedan predmet i da ga stavi u skladište koje se zadaje prilikom stvaranja proizvođača.
- **Potrošač** može da uzme jedan predmet iz skladišta koje se zadaje prilikom stvaranja potrošača.

Rešenje:

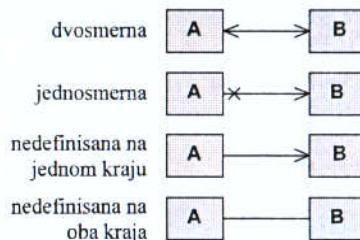
a) Detaljni dijagram



b) Sažeti dijagram



c) Navigabilnost po *UML2* (*StarUML* ne podržava)



d) Kôd koji je na jeziku *Java* generisao *StarUML*

```
//
//
//  Generated by StarUML(tm) Java Add-In
//
//  @ Project : PS
//  @ File Name : Predmet.java
//  @ Date : 16.8.2007
//  @ Author :
```

```
public class Predmet {
    private int id;
    private static int posId = 0;
    public void Predmet() {

    }

    public int dohvId() {

    }
}
```

```
//  @ File Name : Skladiste.java
public class Skladiste {
    private int kap;
    private int ulaz;
    private int izlaz;
    private int duz;
    private Predmet predmeti;
    public void Skladiste(int kap) { }
    public int duz() { }
    public boolean pun() { }
    public boolean prazan() { }
    public void stavi(Predmet p) { }
    public Predmet uzmi() { }
}
```

```
//  @ File Name : Proizvodjac.java
public class Proizvodjac {
    private Skladiste sklad;
    public void Proizvodjac(Skladiste s) { }
    public void proizvedi() { }
}
```

```
//  @ File Name : Potrosac.java
public class Potrosac {
    private Skladiste sklad;
    public void Potrosac(Skladiste s) { }
    public Predmet potrosi() { }
}
```

e) Kôd koji je na jeziku C# generisao *StarUML*

```
//
//
//  Generated by StarUML(tm) C# Add-In
//
//  @ Project : PS
//  @ File Name : Predmet.cs
//  @ Date : 16.8.2007
//  @ Author :
//
//

public class Predmet {
    private int id ;
    private static int posId = 0;
    public Predmet(){
    }
    public int dohvId(){
    }
}
```

```
//  @ File Name : Skladiste.cs
public class Skladiste {
    private int kap ;
    private int ulaz ;
    private int izlaz ;
    private int duz ;
    private Predmet predmeti;
    public Skladiste(int kap){ }
    public int duz(){ }
    public boolean pun(){ }
    public boolean prazan(){ }
    public void stavi(Predmet p){ }
    public Predmet uzmi(){ }
}
```

```
//  @ File Name : Proizvodjac.cs
public class Proizvodjac {
    private Skladiste sklad;
    public Proizvodjac(Skladiste s){ }
    public void proizvedi(){ }
}
```

```
//  @ File Name : Potrosac.cs
public class Potrosac {
    private Skladiste sklad;
    public Potrosac(Skladiste s){ }
    public Predmet potrosi(){ }
}
```


f) Kôd koji je na jeziku C++ generisao StarUML

```
//
//
//  Generated by StarUML(tm) C++ Add-In
//
//  @ Project : PS
//  @ File Name : Predmet.h
//  @ Date : 16.8.2007
//  @ Author :
//
//

#if !defined( _PREDMET_H )
#define _PREDMET_H

class Predmet {
public:
    Predmet();
    int dohvId();
private:
    int id;
    static int posId = 0;
};

#endif // _PREDMET_H
```

```
//  @ File Name : Predmet.cpp
#include "Predmet.h"
Predmet::Predmet() { }
int Predmet::dohvId() { }
```

```
//  @ File Name : Skladiste.h
#if !defined( SKLADISTE_H )
#define _SKLADISTE_H
#include "Predmet.h"
class Skladiste {
public:
    Skladiste(int kap);
    int duz();
    boolean pun();
    boolean prazan();
    void stavi(Predmet p);
    Predmet uzmi();
private:
    int kap;
    int ulaz;
    int izlaz;
    int duz;
    Predmet *predmeti;
};
#endif // _SKLADISTE_H
```

```
//  @ File Name : Proizvodjac.h
#if !defined( PROIZVODJAC_H )
#define _PROIZVODJAC_H
#include "Skladiste.h"
class Proizvodjac {
public:
    Proizvodjac(Skladiste s);
    void proizvedi();
private:
    Skladiste *sklad;
};
#endif // _PROIZVODJAC_H
```

```
//  @ File Name : Skladiste.cpp
#include "Skladiste.h"
#include "Predmet.h"
Skladiste::Skladiste(int kap) { }
int Skladiste::duz() { }
boolean Skladiste::pun() { }
boolean Skladiste::prazan() { }
void Skladiste::stavi(Predmet p) { }
Predmet Skladiste::uzmi() { }
```

```
//  @ File Name : Potrosac.h
#if !defined( POTROSAC_H )
#define _POTROSAC_H
#include "Skladiste.h"
#include "Predmet.h"
class Potrosac {
public:
    Potrosac(Skladiste s);
    Predmet potrosi();
private:
    Skladiste *sklad;
};
#endif // _POTROSAC_H
```

```
//  @ File Name : Proizvodjac.cpp
#include "Proizvodjac.h"
#include "Skladiste.h"
Proizvodjac::Proizvodjac(Skladiste s){}
void Proizvodjac::proizvedi() { }
```

```
//  @ File Name : Potrosac.cpp
#include "Potrosac.h"
#include "Skladiste.h"
#include "Predmet.h"
Potrosac::Potrosac(Skladiste s) { }
Predmet Potrosac::potrosi() { }
```

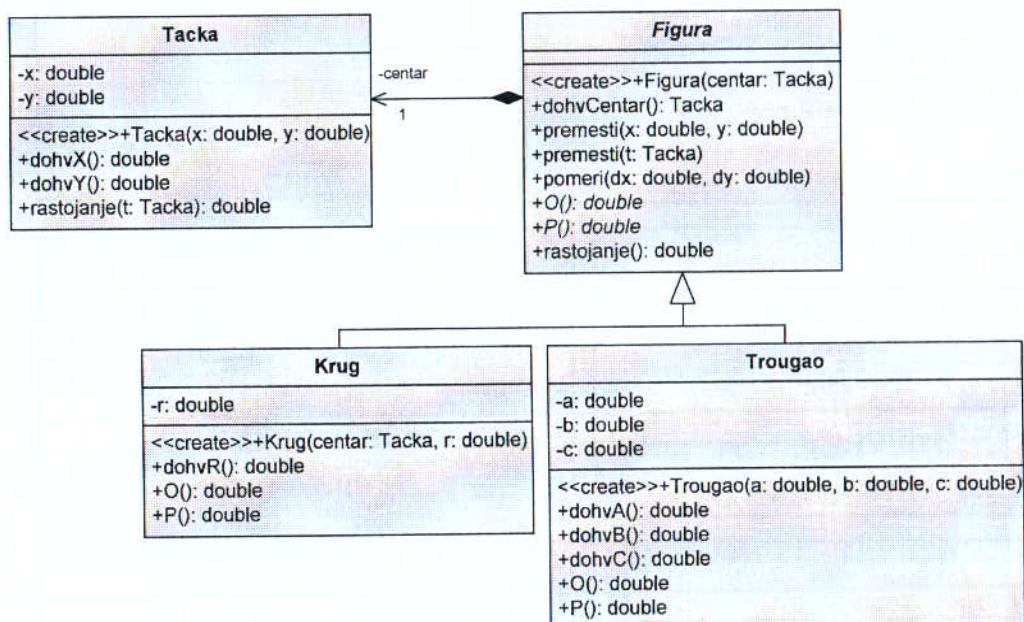
Zadatak 2 Tačka i figure u ravni (nasleđivanje i apstraktne klase)

Nacrtati dijagram klasa na jeziku *UML* sledećeg sistema klasa:

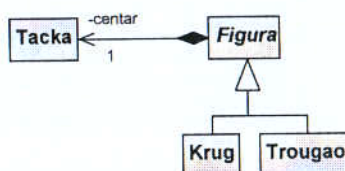
- **Tačka** u ravni zadaje se realnim koordinatama koje mogu da se dohvate. Može da se izračuna rastojanje do zadate tačke.
- Apstraktna **figura** u ravni zadaje se tačkom koja predstavlja centar figure i koja može da se dohvati. Figura može da se premesti na novo mesto i da se pomeri za zadati pomak duž koordinatnih osa. Može da se izračuna obim i površina figure i da se odredi rastojanje od centra figure do centra zadate figure.
- **Krug** u ravni je figura zadata poluprečnikom koji može da se dohvati.
- **Trougao** u ravni je figura zadata dužinama stranica koje mogu da se dohvate.

Rešenje:

a) Detaljni dijagram



b) Sažeti dijagram



c) Kôd koji je na jeziku *Java* generisao *StarUML*

```
// @ File Name : Tacka.java
public class Tacka {
    private double x;
    private double y;
    public void Tacka(double x, double y) { }
    public double dohvX() { }
    public double dohvY() { }
    public double rastojanje(Tacka t) { }
}
```

```
// @ File Name : Figura.java
public abstract class Figura {
    private Tacka centar;
    public void Figura(Tacka centar) { }
    public Tacka dohvCentar() { }
    public void preмести(double x, double y) { }
    public void preмести(Tacka t) { }
    public void pomeri(double dx, double dy) { }
    public abstract double O();
    public abstract double P();
    public double rastojanje() { }
}
```

```
// @ File Name : Krug.java
public class Krug extends Figura {
    private double r;
    public void Krug(Tacka centar, double r) { }
    public double dohvR() { }
    public double O() { }
    public double P() { }
    public double O() { }
    public double P() { }
}
```

```
// @ File Name : Trougao.java
public class Trougao extends Figura {
    private double a;
    private double b;
    private double c;
    public void Trougao(double a, double b, double c) { }
    public double dohvA() { }
    public double dohvB() { }
    public double dohvC() { }
    public double O() { }
    public double P() { }
    public double O() { }
    public double P() { }
}
```

d) Kôd koji je na jeziku C# generisao *StarUML*

```
// @ File Name : Tacka.cs
public class Tacka {
    private double x ;
    private double y ;
    public Tacka(double x, double y){ }
    public double dohvX(){ }
    public double dohvY(){ }
    public double rastojanje(Tacka t){ }
}
```

```
// @ File Name : Figura.cs
public abstract class Figura {
    private Tacka centar;
    public Figura(Tacka centar){ }
    public Tacka dohvCentar(){ }
    public void premesti(double x, double y){ }
    public void premesti(Tacka t){ }
    public void pomeri(double dx, double dy){ }
    public abstract double O();
    public abstract double P();
    public double rastojanje(){ }
}
```

```
// @ File Name : Krug.cs
public class Krug : Figura{
    private double r ;
    public Krug(Tacka centar, double r){ }
    public double dohvR(){ }
    public double O(){ }
    public double P(){ }
}
```

```
// @ File Name : Trougao.cs
public class Trougao : Figura{
    private double a ;
    private double b ;
    private double c ;
    public Trougao(double a, double b, double c){ }
    public double dohvA(){ }
    public double dohvB(){ }
    public double dohvC(){ }
    public double O(){ }
    public double P(){ }
}
```


e) Kôd koji je na jeziku C++ generisao StarUML

```
// @ File Name : Tacka.h
#ifndef TACKA_H
#define TACKA_H
class Tacka {
public:
    Tacka(double x, double y);
    double dohvX();
    double dohvY();
    double rastojanje(Tacka t);
private:
    double x;
    double y;
};
#endif // _TACKA_H
```

```
// @ File Name : Tacka.cpp
#include "Tacka.h"
Tacka::Tacka(double x, double y) { }
double Tacka::dohvX() { }
double Tacka::dohvY() { }
double Tacka::rastojanje(Tacka t) { }
```

```
// @ File Name : Figura.h
#ifndef FIGURA_H
#define FIGURA_H
#include "Tacka.h"
class Figura {
public:
    Figura(Tacka centar);
    Tacka dohvCentar();
    void premesti(double x, double y);
    void premesti(Tacka t);
    void pomeri(double dx, double dy);
    virtual double O() = 0;
    virtual double P() = 0;
    double rastojanje();
private:
    Tacka centar;
};
#endif // _FIGURA_H
```

```
// @ File Name : Figura.cpp
#include "Figura.h"
#include "Tacka.h"
Figura::Figura(Tacka centar) { }
Tacka Figura::dohvCentar() { }
void Figura::premesti(double x, double y) { }
void Figura::premesti(Tacka t) { }
void Figura::pomeri(double dx, double dy) { }
double Figura::rastojanje() { }
```

```
// @ File Name : Krug.h
#ifndef KRUG_H
#define KRUG_H
#include "Figura.h"
#include "Tacka.h"

class Krug : public Figura {
public:
    Krug(Tacka centar, double r);
    double dohvR();
    double O();
    double P();
double O();
double P();
private:
    double r;
};
#endif // _KRUG_H
```

```
// @ File Name : Krug.cpp
#include "Krug.h"
#include "Tacka.h"
Krug::Krug(Tacka centar, double r) { }
double Krug::dohvR() { }
double Krug::O() { }
double Krug::P() { }
double Krug::O() { }
double Krug::P() { }
```

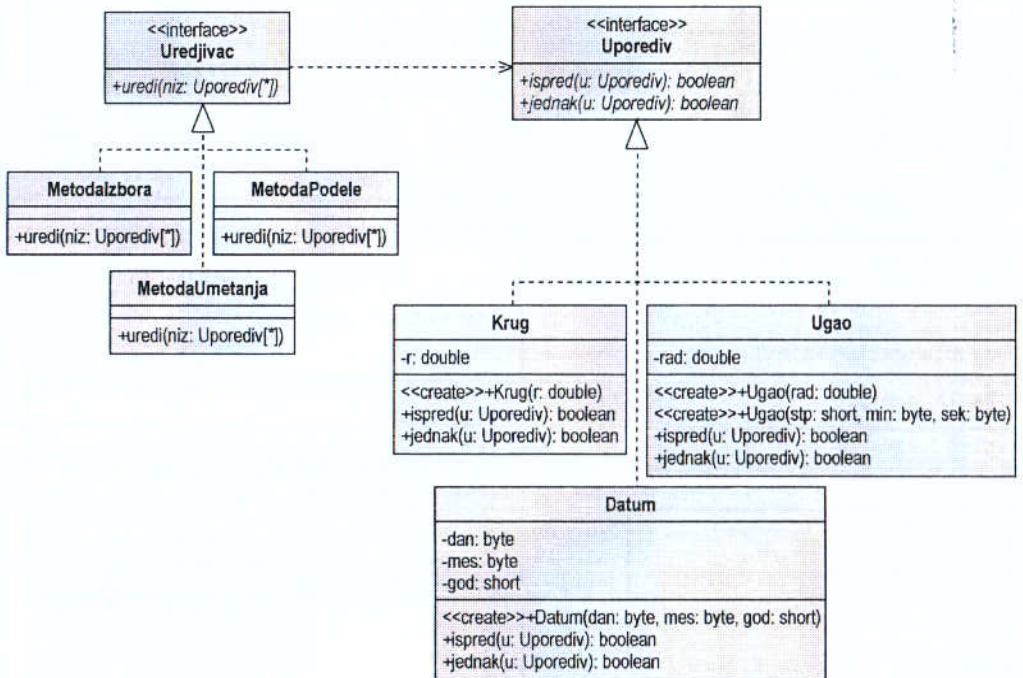
Zadatak 3 Uređivači uporedivih stvari (interfejsi)

Nacrtati dijagram klasa na jeziku *UML* sledećeg sistema tipova:

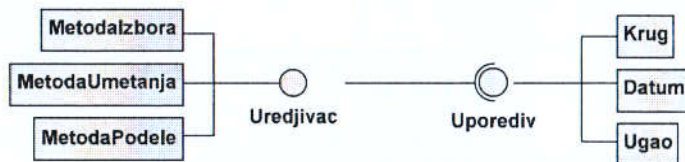
- Za apstraktnu **uporedivu** stvar može da se ispita da li se nalazi ispred i da li je jednaka drugoj uporedivoj stvari.
- **Krug**, **datum** i **ugao** su uporedive stvari.
- Apstraktan **uređivač** može da uredi niz uporedivih stvari.
- **Metoda izbora**, **metoda umetanja** i **metoda podele** su uređivači.

Rešenje:

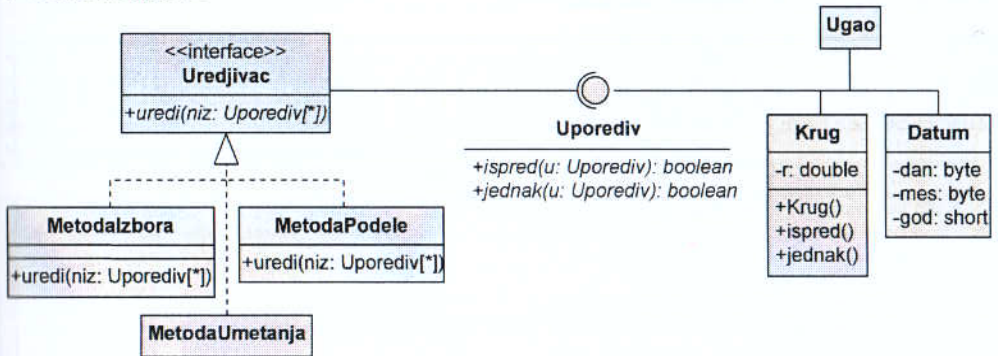
a) Detaljni dijagram



b) Sažeti dijagram



c) Mešoviti dijagram

d) Kôd koji je na jeziku *Java* generisao *StarUML*

```
// @ File Name : Uporediv.java
public interface Uporediv {
    public abstract boolean ispred(Uporediv u);
    public abstract boolean jednak(Uporediv u);
}
```

```
// @ File Name : Krug.java
public class Krug implements Uporediv {
    private double r;
    public void Krug(double r) { }
    public boolean ispred(Uporediv u) { }
    public boolean jednak(Uporediv u) { }
    public boolean ispred(Uporediv u){
    public boolean jednak(Uporediv u){
}
```

```
// @ File Name : Uredjivac.java
public interface Uredjivac {
    public abstract void uredi(Uporediv[] niz);
}
```

```
// @ File Name : MetodaIzbora.java
public class MetodaIzbora implements Uredjivac {
    public void uredi(Uporediv[] niz) { }
    public void uredi(Uporediv[] niz){
}
```

e) Kôd koji je na jeziku C# generisao *StarUML*

```
// @ File Name : Uporediv.cs
public interface Uporediv {
    abstract boolean ispred(Uporediv u);
    abstract boolean jednak(Uporediv u);
}
```

```
// @ File Name : Krug.cs
public class Krug : Uporediv{
    private double r ;
    public Krug(double r){ }
    public boolean ispred(Uporediv u){ }
    public boolean jednak(Uporediv u){ }
}
```

```
// @ File Name : Uredjivac.cs
public interface Uredjivac {
    abstract void uredi(Uporediv[] niz);
}
```

```
// @ File Name : MetodaIzbora.cs
public class MetodaIzbora : Uredjivac{
    public void uredi(Uporediv[] niz){ }
}
```

f) Kôd koji je na jeziku C++ generisao *StarUML*

```
// @ File Name : Uporediv.h
#ifndef _UPOREDIV_H
#define _UPOREDIV_H
class Uporediv {
public:
    virtual boolean ispred(Uporediv u) = 0;
    virtual boolean jednak(Uporediv u) = 0;
};
#endif // _UPOREDIV_H
```

```
// @ File Name : Krug.h
#ifndef _KRUG_H
#define _KRUG_H
#include "Uporediv.h"
class Krug : public Uporediv {
public:
    Krug(double r);
    boolean ispred(Uporediv u);
    boolean jednak(Uporediv u);
    boolean ispred(Uporediv u);
    boolean jednak(Uporediv u);
private:
    double r;
};
#endif // _KRUG_H
```



```
// @ File Name : Krug.cpp
#include "Krug.h"
#include "Uporediv.h"
void Krug::Krug(double r) { }
boolean Krug::ispred(Uporediv u) { }
boolean Krug::jednak(Uporediv u) { }
boolean Krug::ispred(Uporediv u) { }
boolean Krug::jednak(Uporediv u) { }
```

```
// @ File Name : Uredjivac.h
#if !defined(_UREDJIVAC_H)
#define _UREDJIVAC_H
class Uredjivac {
public:
    virtual void uredi(Uporediv[] niz) = 0;
};
#endif // _UREDJIVAC_H
```

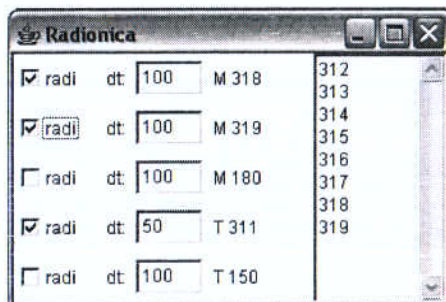
```
// @ File Name : MetodaIzbora.h
#if !defined(_METODAIZBORA_H)
#define _METODAIZBORA_H
#include "Uredjivac.h"
class MetodaIzbora : public Uredjivac {
public:
    void uredi(Uporediv[] niz);
    void uredi(Uporediv[] niz);
};
#endif // _METODAIZBORA_H
```

```
// @ File Name : MetodaIzbora.cpp
#include "MetodaIzbora.h"
void MetodaIzbora::uredi(Uporediv[] niz) { }
void MetodaIzbora::uredi(Uporediv[] niz) { }
```

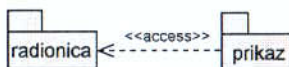
Zadatak 4 Proizvod, skladište, mehanizmi i motor (dijagram paketa: aktivna klasa)

Nacrtati dijagram paketa i dijagram klasa na jeziku *UML* sledećeg sistema tipova:

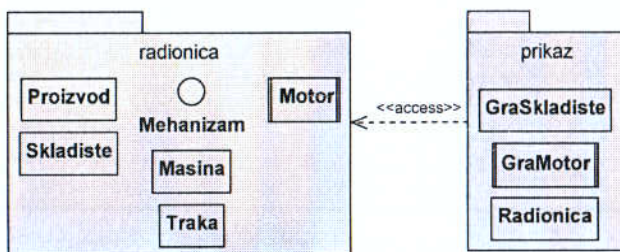
- **Proizvod** ima jednoznačan, automatski generisan celobrojan identifikator. Tekstualni opis proizvoda sadrži samo njegov identifikator.
- **Skladište** proizvoda može da smesti zadati broj proizvoda. Stvara se prazno, posle čega može da se dodaje i uzima po jedan proizvod. Pri pokušaju stavljanja u puno skladište, odnosno uzimanja iz praznog skladišta, nit izvršioca radnje privremeno se blokira. Tekstualni opis skladišta sastoji se od niza tekstualnih opisa sadržanih proizvoda, jedan proizvod po redu.
- Apstraktan **mehanizam** predviđa izvođenje neke radnje.
- **Mašina** je mehanizam čija se radnja sastoji od stvaranja jednog proizvoda i njegovog stavljanja u skladište koje je zadato prilikom stvaranja mašine. Tekstualni opis sadrži slovo **M** i tekstualni opis proizvoda koji je upravo stavljen u skladište.
- Pokretna **traka** je mehanizam čija se radnja sastoji od uzimanja jednog proizvoda iz skladišta koje je zadato prilikom stvaranja trake. Tekstualni opis sadrži slovo **T** i tekstualni opis proizvoda koji je upravo uzet iz skladišta.
- Aktivni **motor** ponavlja radnju jednog mehanizma koji se zadaje prilikom stvaranja motora u slučajnim vremenskim intervalima $\Delta t \pm 20\%$ ms (na početku je $\Delta t = 100$ ms i može da se promeni za vreme života motora). Rad motora može da se zaustavi, da se nastavi dalje i da se prekine.
- **Grafičko skladište** je skladište koje se inicijalizuje grafičkim tekstualnim prostorom (TextArea) na kome prikazuje svaku promenu stanja skladišta.
- **Grafički motor** je motor koji se inicijalizuje grafičkom pločom (Panel) koju popunjava komponentama tako da korisnik može zaustaviti i nastaviti rad motora, promeniti vremenski interval Δt (promena ima efekta odmah) i može da se prikaže tekstualni opis pridruženog mehanizma.
- **Radionica** je program koji na grafičkoj korisničkoj površi prema slici prikazuje rad s jednim skladištem kapaciteta 20, tri mašine i dve pokretne trake.

**Rešenje:**

a) Dijagram paketa



b) Dijagram paketa sa sadržajem



e) Kôd koji je na jeziku *Java* generisao *StarUML*

- paketi su stavljani u odgovarajuće potfascikle
- klase iz drugih paketa koriste se punim imenom

```
// @ File Name : Proizvod.java
package radionica;
public class Proizvod {
    private static int posId = 0;
    private int id = ++posId;
    public String toString() { }
}
```

```
// @ File Name : Radionica.java
package prikaz;
public class Radionica {
    private static final int BR_M = 3;
    private static final int BR_T = 2;
    private GraMotor motori;
    private GraSkladiste sklad;
    public void Radionica() { }
    private void popuniProzor() { }
    public static void main(String[] varg) { }
}
```

```
// @ File Name : Skladiste.java
package radionica;
public class Skladiste {
    private int ualz;
    private int izlaz;
    private int duz;
    private Proizvod niz;
    public void Skladiste(int kap) { }
    public void stavi(Proizvod pro) { }
    public Proizvod uzmi() { }
    public String toString() { }
}
```

```
// @ File Name : GraSkladiste.java
package prikaz;
public class GraSkladiste extends radionica.Skladiste {
    private TextArea prikaz;
    public void GraSkladiste(int kap, TextArea prikaz) { }
    public void stavi(radionica.Proizvod p) { }
    public radionica.Proizvod uzmi() { }
}
```

f) Kôd koji je na jeziku *C#* generisao *StarUML*

- paketi su stavljani u odvojene prostore imena i potfascikle
- upotreba klasa iz drugog paketa nije korektna

```
// @ File Name : Proizvod.cs
namespace radionica{
    public class Proizvod {
        private static int posId = 0;
        private int id = ++posId;
        public String toString(){ }
    }
}
```

```
// @ File Name : Radionica.cs
namespace prikaz{
    public class Radionica {
        private static readonly int BR_M = 3;
        private static readonly int BR_T = 2;
        private GraMotor motori;
        private GraSkladiste sklad;
        public Radionica(){ }
        private void popuniProzor(){ }
        public static void
            main(String[] varg){ }
    }
}
```

```
// @ File Name : Skladiste.cs
namespace radionica{
    public class Skladiste {
        private int ualz ;
        private int izlaz ;
        private int duz ;
        private Proizvod niz;
        public Skladiste(int kap){ }
        public void stavi(Proizvod pro){ }
        public Proizvod uzmi(){ }
        public String toString(){ }
    }
}
```

```
// @ File Name : GraSkladiste.cs
namespace prikaz{
    public class GraSkladiste : Skladiste{
        private TextArea prikaz ;
        public GraSkladiste(int kap, TextArea prikaz){ }
        public void stavi(Proizvod p){ }
        public Proizvod uzmi(){ }
    }
}
```


g) Kôd koji je na jeziku C++ generisao *StarUML*

- paketi su na zahtev stavljeni u odvojene prostore imena i u istu fasciklu
- upotreba klasa iz drugog paketa nije korektna

```
// @ File Name : Proizvod.h
#ifndef _PROIZVOD_H
#define _PROIZVOD_H
namespace radionica {
    class Proizvod {
    public:
        String toString();
    private:
        static int posId = 0;
        int id = ++posId;
    };
}
#endif // _PROIZVOD_H
```

```
// @ File Name : Proizvod.cpp
#include "Proizvod.h"
namespace radionica {
    String Proizvod::toString() { }
}
```

```
// @ File Name : Skladiste.cpp
#include "Skladiste.h"
#include "Proizvod.h"
namespace radionica {
    Skladiste::Skladiste(int kap) { }
    void Skladiste::stavi(Proizvod pro) { }
    Proizvod Skladiste::uzmi() { }
    String Skladiste::toString() { }
}
```

```
// @ File Name : Skladiste.h
#ifndef _SKLADISTE_H
#define _SKLADISTE_H
#include "Proizvod.h"
namespace radionica {
    class Skladiste {
    public:
        Skladiste(int kap);
        void stavi(Proizvod pro);
        Proizvod uzmi();
        String toString();
    private:
        int ualz;
        int izlaz;
        int duz;
        Proizvod *niz;
    };
}
#endif // _SKLADISTE_H
```

```
// @ File Name : GraSkladiste.h
#ifndef _GRASKLADISTE_H
#define _GRASKLADISTE_H
#include "Skladiste.h"
#include "Proizvod.h"
namespace prikaz {
    class GraSkladiste : public Skladiste {
    public:
        GraSkladiste(int kap, TextArea prikaz);
        void stavi(Proizvod p);
        Proizvod uzmi();
    private:
        TextArea prikaz;
    };
}
#endif // _GRASKLADISTE_H
```

```
// @ File Name : GraSkladiste.cpp
#include "GraSkladiste.h"
#include "Proizvod.h"
namespace prikaz {
    GraSkladiste::GraSkladiste(int kap,
        TextArea prikaz) { }
    void GraSkladiste::stavi(Proizvod p) { }
    Proizvod GraSkladiste::uzmi() { }
}
```

```
// @ File Name : Radionica.h
#ifndef _RADIONICA_H
#define _RADIONICA_H
#include "GraMotor.h"
#include "GraSkladiste.h"
namespace prikaz {
    class Radionica {
    public:
        Radionica();
        static void main(String[] varg);
    private:
        static const int BR_M = 3;
        static const int BR_T = 2;
        void popuniProzor();
        GraMotor motori;
        GraSkladiste sklad;
    };
}
#endif // _RADIONICA_H
```

```
// @ File Name : Radionica.cpp
#include "Radionica.h"
namespace prikaz {
    Radionica::Radionica() { }
    void Radionica::main(String[] varg) { }
    void Radionica::popuniProzor() { }
}
```


Zadatak 5 Samoposluga (generisanje modela na osnovu programa na jeziku Java)

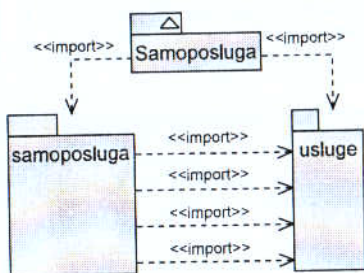
Nacrtati dijagram paketa i dijagrame klasa za sledeći gotov sistem klasa napisan na jeziku *Java* (videti zadatak 9.17 u zbirci zadataka [6]) koji simulira rad samoposluge:

- Aktivan **ulaz** samoposluge ima jedinstven, automatski generisan celobrojan identifikator koji može da se dohvati. Ulaz može da se otvori, zatvori i uništi. Kada je ulaz otvoren kupci ulaze u slučajnim vremenskim intervalima. Najduže vreme između ulazaka kupaca je parametar ulaza. Prikaz ulaza u polju grafičke korisničke površi sadrži identifikator ulaza. Ako je ulaz otvoren prikaz sadrži i identifikator kupca koji je poslednji ušao.
- Aktivan **kupac** ima jedinstven, automatski generisan celobrojan identifikator koji može da se dohvati. Po ulasku u samoposlugu neko slučajno vreme bira robu (najduže vreme je parametar kupca), potom stane u red ispred slučajno odabrane kase. Posle plaćanja napušta samoposlugu.
- **Red** kupaca ispred kase je neograničenog kapaciteta. Kupci se dodaju na kraj reda, a uzimaju sa početka reda. Prikaz reda u polju grafičke korisničke površi sadrži identifikatore kupaca u redu.
- Aktivna **kasa** ima jedinstven, automatski generisan, celobrojan identifikator koji može da se dohvati. Ispred kase postoji jedan red kupaca. Kasa može da se otvori, zatvori i uništi. Kada je kasa otvorena naplaćuje od kupaca koji stoje u redu ispred nje. Naplaćivanje traje slučajno vreme. Najduže vreme naplaćivanja je parametar kase. Prikaz kase u polju grafičke korisničke površi sadrži identifikator kase. Ako je naplaćivanje u toku, prikaz sadrži i identifikator kupca koji se opslužuje.
- **Samoposluga** ima jedinstven, automatski generisan, celobrojan identifikator koji može da se dohvati. U samoposluzi postoji zadati broj ulaza i kasa. U početku ulazi su zatvoreni, a kase su spremne za rad. Samoposluga može da se otvori, zatvori i uništi. Pri otvaranju svi ulazi odmah se otvore. Pri zatvaranju svi ulazi odmah se zatvore. Zatvaranje se smatra završenim kada i poslednji kupac napusti samoposlugu. Prikaz samoposluge u polju grafičke korisničke površi sadrži broj kupaca u samoposluzi.

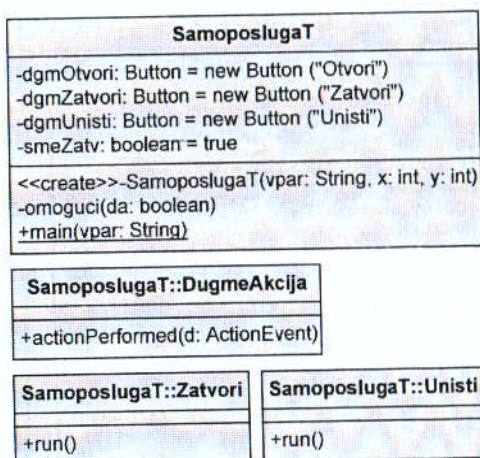
Rešenje:

Sledeći model je generisan pomoću *StarUML*.

a) Dijagram paketa



b) Klase u bezimenom paketu



c) Klase u paketu samoposluga

samoposluga::Ulaz	
-posId: int = 0 -id: int = ++posId -maxUlaz: int -maxKupovina: int -radi: boolean = false	
<<create>>~Ulaz(samop: Samoposluga, maxUl: int, maxKup: int, p: Polje) +run() ~zatvori() ~otvori() ~unisti() ~id(): int	

samoposluga::Kupac	
-posId: int = 0 -id: int = ++posId -maxKupovina: int	
<<create>>~Kupac(samop: Samoposluga, maxKup: int) +run() ~naplaceno() ~id(): int	

samoposluga::Kasa	
-posId: int = 0 -id: int = ++posId -maxNaplat: int -radi: boolean = true	
<<create>>~Kasa(samop: Samoposluga, maxNap: int, p: Polje, pRed: Polje) +run() ~zatvori() ~otvori() ~unisti() ~staviURed(kupac: Kupac) ~id(): int	

samoposluga::Red	
<<create>>~Red(p: Polje) ~stavi(kupac: Kupac) ~uzmi(): Kupac {JavaThrows = InterruptedException} +toString(): String	

Red::Elem	
<<create>>~Elem(k: Kupac)	

samoposluga::Samoposluga	
-posId: int = 0 -id: int = ++posId -brojKupaca: int = 0 -radi: boolean = false	
<<create>>+Samoposluga(brUlaza: int, brKasa: int, maxUlaz: int, maxKupovina: int, maxNaplat: int, p: Polje, pUlazi: Polje, pKase: Polje, pRedovi: Polje) +otvori() +zatvori() ~usaoKupac() ~izasaoKupac() +id(): int ~dohvatiKase(): Kasa +unisti()	

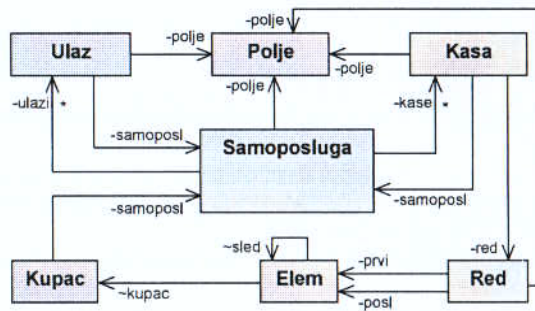
d) Klase u paketu usluge

usluge::Polje
-komp: Component -ind: int -prozor: Component
<<create>>+Polje(k: Component, i: int) {JavaThrows = GKompNeOdgovara} <<create>>+Polje(k: Component) {JavaThrows = GKompNeOdgovara} +komponenta(): Component +indeks(): int +pisi(tekst: String) +pisi(broj: long) +pisi(broj: long, baza: int) +pisi(broj: double) +pisi(b: boolean) +pisi(znak: char) +pisi(o: Object) +String(): String +Long(): long +Long(baza: int): long +Int(): int +Int(baza: int): int +Short(): short +Short(baza: int): short +Byte(): byte +Byte(baza: int): byte +Double(): double +Float(): float +Boolean(): boolean +Char(): char

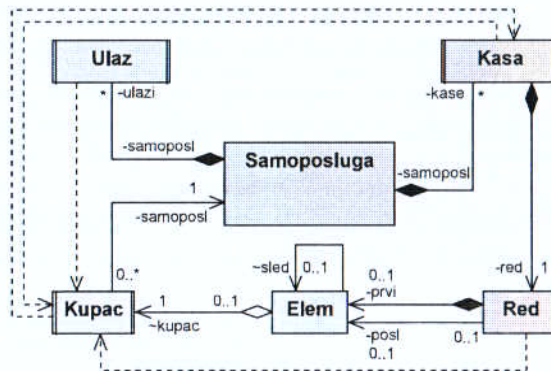
usluge::Greska
-por:
<<create>>+Greska() <<create>>+Greska(p: String) +toString(): String #imaPoruke(): boolean

usluge::GKompNeOdgovar
<<create>>+GKompNeOdgovara(k: Component)

e) Generisani dijagram klasa



f) Poboljšani dijagram klasa (bez klase Polje)



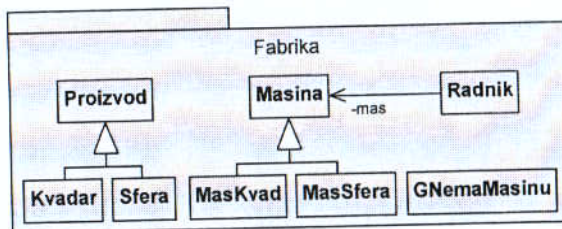
Zadatak 6 Proizvodi, mašine i radnik (generisanje modela na osnovu programa na jeziku C++)

Nacrtati dijagram paketa i dijagrame klasa za sledeći gotov sistem klasa napisan na jeziku C++ (videti zadatak 5.9 u zbirci zadataka [7]):

- Apstraktan **proizvod** ima jednoslovnu oznaku vrste i jedinstven, automatski generisan celobrojan identifikator. Može da se dohvati vrsta i identifikator proizvoda i da se izračuna zapremina. Pri pisanju u izlazni tok (`it<<p`) piše se vrsta i identifikacioni broj proizvoda.
- Apstraktna **mašina** može da proizvodi apstraktan proizvod i zna se koliko je proizvoda ta mašina proizvela. Može da se dohvati oznaka vrste proizvoda koje data mašina proizvodi i broj proizvedenih proizvoda. Ne može da se stvara kopija mašine.
- **Kvadar** je proizvod zadat dužinama ivica. Oznaka vrste proizvoda je **K**. Pri pisanju navode se i dimenzije kvadra.
- **Sfera** je proizvod zadat poluprečnikom. Oznaka vrste proizvoda je **S**. Pri pisanju navodi se i poluprečnik sfere.
- **Mašina za kvadre** je mašina koja može da proizvodi kvadre zadatih dimenzija. Parametri mašine (dimenzije kvadara koje proizvodi) ne mogu da se promene, ali mogu da se dohvate.
- **Mašina za sfere** je mašina koja može da proizvodi sfere zadatog poluprečnika. Parametar mašine (poluprečnik sfera koje proizvodi) ne može da se promeni, ali može da se dohvati.
- **Radnik** ima ime i jedinstven, automatski generisan celobrojan identifikator. Izrađuje proizvode određene vrste pomoću odgovarajuće mašine. Mašina na kojoj radnik radi može da se promeni (`r+=m`). Zna se koliko je proizvoda izradio na mašini na kojoj trenutno radi. Radnik može da ne bude raspoređen ni na jednu mašinu. U tom slučaju pokušaj izrade proizvoda ili dohvaćanja broja izrađenih proizvoda prijavljuje se izuzetkom tipa specijalne klase. Pri pisanju u izlazni tok (`it<<r`) piše se ime i identifikator radnika. U slučaju da je radnik raspoređen na neku mašinu piše se i vrsta proizvoda koje trenutno izrađuje i broj proizvoda koje je izradio na toj mašini od početka rada na njoj.

Rešenje:

a) Model koji je generisao *StarUML*



Fabrika::Proizvod
-ukld: int
-id: int
<<create>>-Proizvod()
<<create>>-Proizvod(: Proizvod)
<<destroy>>-Proizvod()
<<CppOperator>>+=(: Proizvod): Proizvod
+dohv(): int
+vrsta(): char
+V(): double
#pisi(it: ostream): void
<<CppFriend>>#<<(it: ostream, p: Proizvod): ostream

Fabrika::Masina
-br: int
-pravi(): Proizvod
<<create>>-Masina(: Masina)
<<CppOperator>>=: (Masina): void
<<create>>-Masina()
+vrsta(): char
+napravi(): Proizvod
+broj(): int

Fabrika::Kvadar
-a: double -b: double -c: double
-pisi(it: ostream): void <<create>>-Kvadar(aa: double, bb: double, cc: double) +vrsta(): char +V(): double

Fabrika::Sfera
-r: double
-pisi(it: ostream): void <<create>>-Sfera(rr: double) +vrsta(): char +V(): double

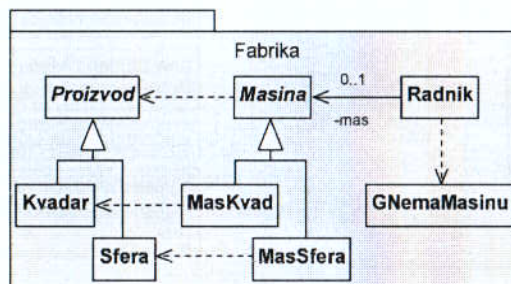
Fabrika::MasKvad
-a: double -b: double -c: double
-pravi(): Kvadar <<create>>-MasKvad(aa: double, bb: double, cc: double) +vrsta(): char +dohvA(): double +dohvB(): double +dohvC(): double

Fabrika::MasSfera
-r: double
-pravi(): Sfera <<create>>-MasSfera(rr: double) +vrsta(): char +dohvR(): double

Fabrika::Radnik
-ukId: int -id: int -ime: char -br: int
<<create>>-Radnik(: Radnik) <<CppOperator>>-=: Radnik): void <<create>>-Radnik(iime: char) <<destroy>>-Radnik() +dohvIme(): char +dohvId(): int <<CppOperator>>+= (m: Masina): Radnik +broj(): int +napravi(): Proizvod <<CppFriend>>+<<(it: ostream, r: Radnik): ostream

Fabrika::GNemaMasinu
-ime: char -id: int
<<CppOperator>>-=: GNemaMasinu): void <<create>>-GNemaMasinu(iime: char, iid: int) <<create>>-GNemaMasinu(g: GNemaMasinu) <<destroy>>-GNemaMasinu() <<CppFriend>>+<<(it: ostream, g: GNemaMasinu): ostream

b) Poboljšani model



Zadatak 7 Pravila, student, studentski odsek (projektni uzorak *Unikat*)

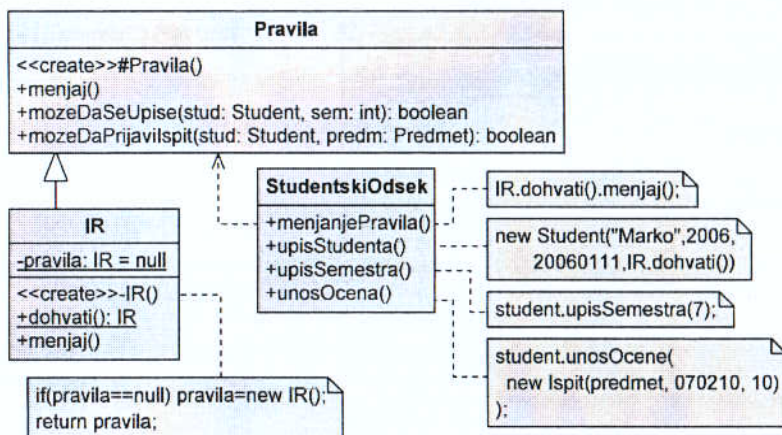
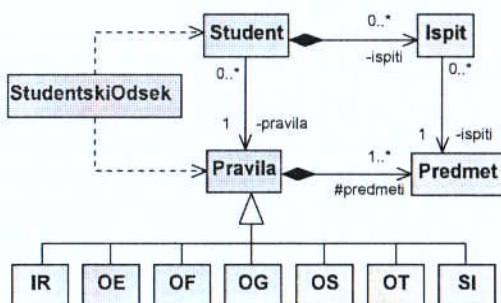
Nacrtati dijagram klasa na jeziku *UML* sledećeg modela fakulteta:

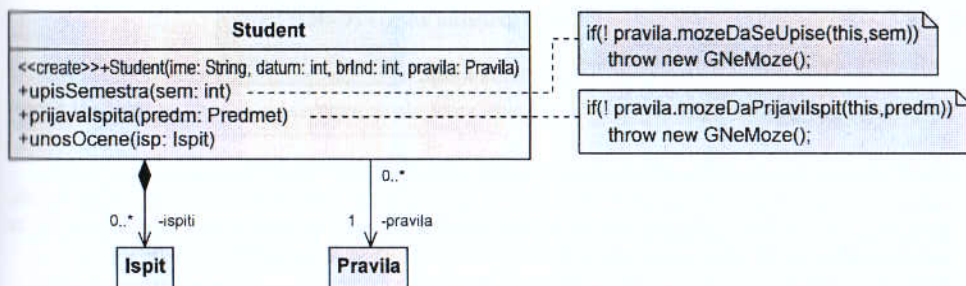
- **Pravila** studiranja određuju uslove pod kojima student može da upiše semestar i da prijavljuje ispite. Pravila mogu da se menjaju. Deo pravila čini skup predmeta iz kojih student treba da polaže ispite.
- Za svaki odsek fakulteta postoje jedinstvena specifična pravila.
- **Student** može da upisuje semestar, da prijavljuje ispite i da mu se unose ocene ispita.
- **Studentski odsek** održava pravila studiranja, upisuje nove studente pridružujući im pravila studiranja zavisno od odseka, upisuje studente u naredne semestre i unosi ocene ispita.

Koristiti projektni uzorak *Unikat* za pravila studiranja.

Rešenje:

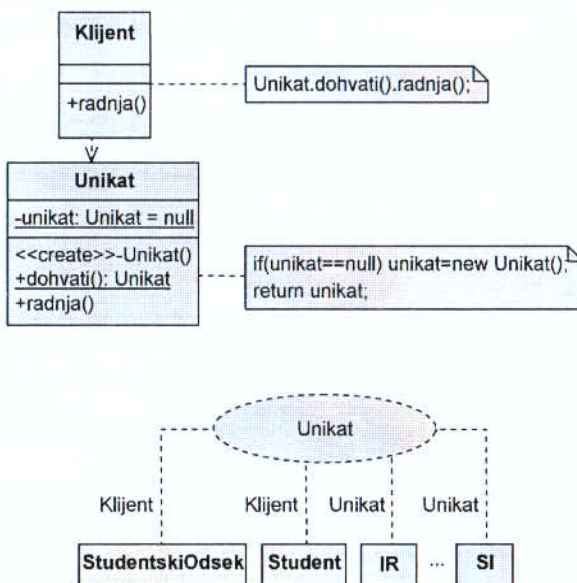
a) Dijagrami klasa





b) Projektni uzorak **Unikat** (Singleton)

- objektni uzorak stvaranja
- obezbeđuje da postoji samo jedan primerak klase čije se stvaranje odlaže do prvog pristupa objektu
- klijent dohvata unikatni objekat isključivo pristupnom metodom



- Alternativno rešenje:
 - samo statički atributi i metode
 - ne pravi se nijedan objekat
 - privatan konstruktor koga niko ne poziva
 - ne može da se obezbedi polimorfno ponašanje

Zadatak 8 Jednostavni predmeti, sklopovi i radnici (projektni uzorci **Prototip** i **Sastav**)

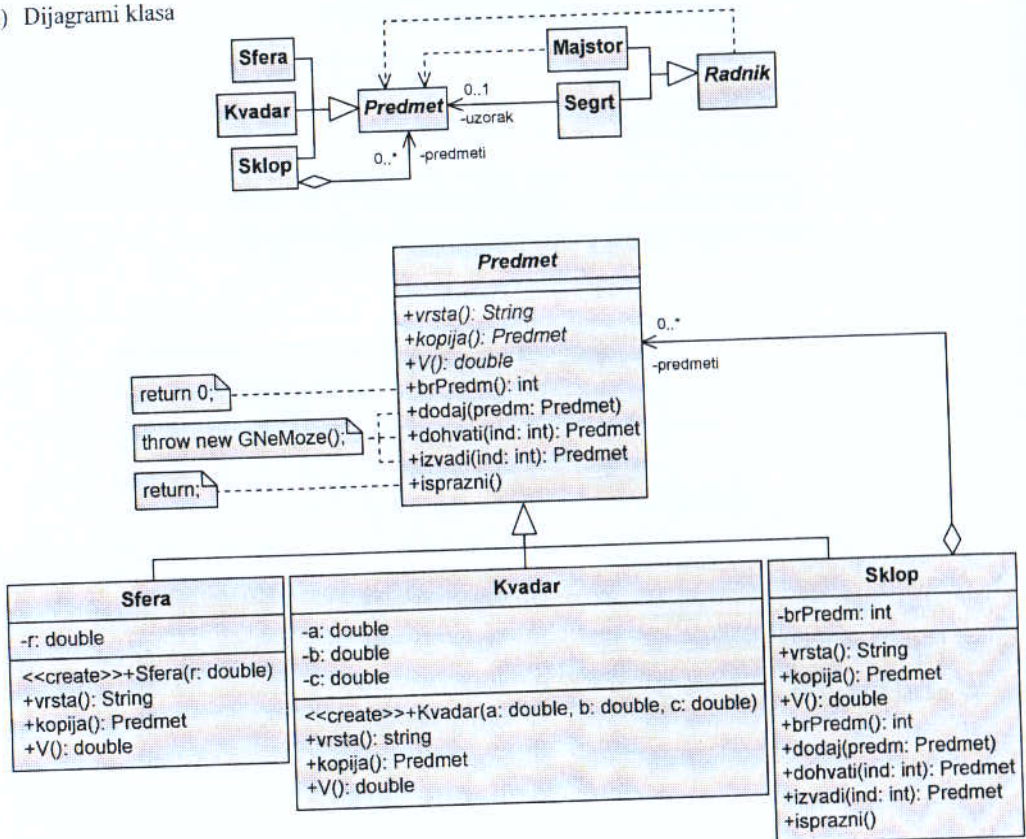
Nacrtati dijagram klasa na jeziku *UML* sledećeg sistema klasa:

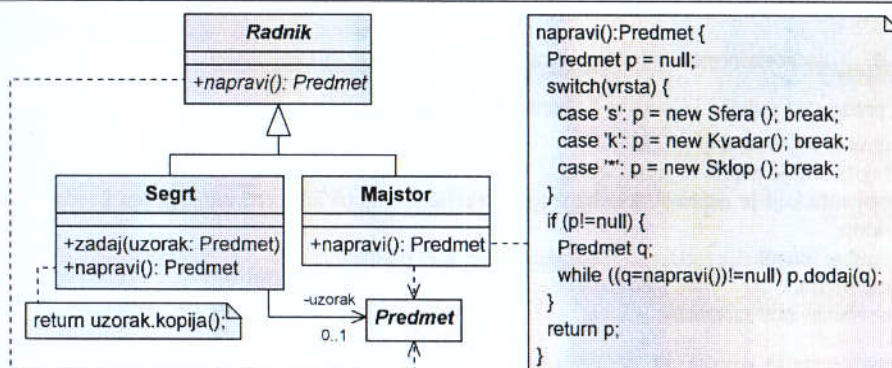
- Apstraktnom *predmetu* može da se dohvati naziv vrste, da se napravi kopija i da se izračuna zapremina.
- *Sfera* i *kvadar* su predmeti zadati poluprečnikom, odnosno dužinama ivica.
- *Sklop* je predmet koji može da sadrži proizvoljan broj predmeta proizvoljne vrste. Stvara se prazan, posle čega se predmeti dodaju jedan po jedan. Može da se dohvati broj predmeta u sklopu, da se dohvati i da se izvadi predmet sa zadatim rednim brojem i da se sklop isprazni.
- Apstraktan *radnik* može da napravi jedan predmet.
- *Šegrt* je radnik koji predmete pravi kao verne kopije drugog predmeta koji mu se daje kao uzorak. Uzorak može da se zameni drugim predmetom.
- *Majstor* je radnik koji može da napravi predmet po svom nahodenju.

Koristiti projektne uzorke **Prototip** i **Sastav**.

Rešenje:

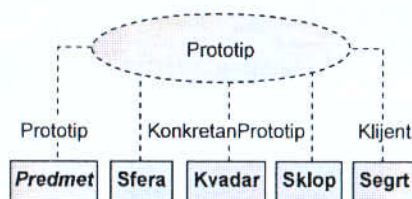
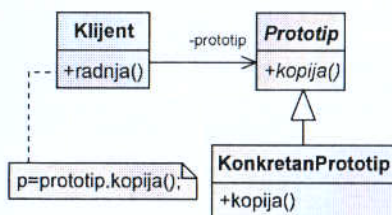
a) Dijagrami klasa





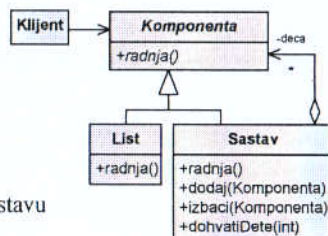
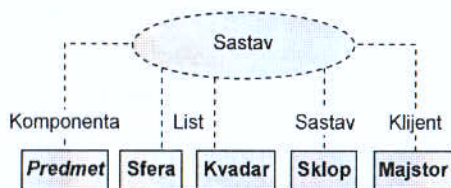
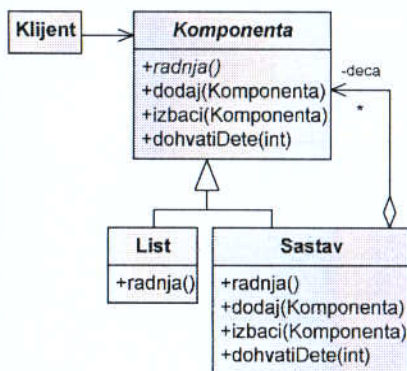
b) Projektni uzorak **Prototip** (*Prototype*)

- objektni uzorak stvaranja
- polimorfno kopiranje objekata na osnovu zadatog originala
- klijent traži od originala da napravi svoju kopiju



c) Projektni uzorak **Sastav** (*Composite*)

- objektni uzorak strukture
- hijerarhijska struktura delova i sklopova čiji elementi mogu da se obrađuju uniformno, nezavisno da li su čvorovi ili listovi
- klijent objekte koristi kroz zajednički interfejs komponente



- Alternativno rešenje
 - rukovanje sastavom ne nalazi se u komponenti već samo u sastavu
 - mana: nije isti interfejs za listove i za sastav

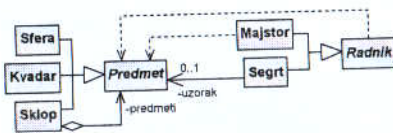
Zadatak 9 Jednostavni predmeti, sklopovi i radnici (dijagrami objekata i interakcije)

Za model predmeta i radnika iz zadatka 8 nacrtati na jeziku UML:

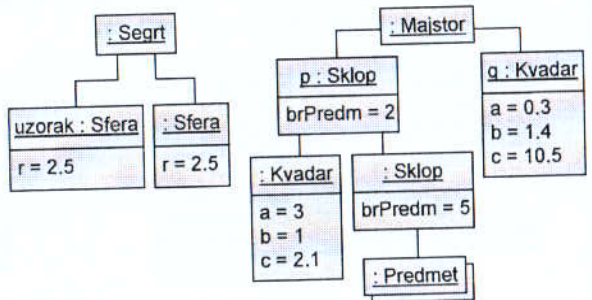
- dijagrame objekata koji prikazuju:
 - šegrtu koji je napravio jednu sferu,
 - majstora koji je napravio kvadar u toku pravljenja sklopa koji već sadrži jedan kvadar i jedan sklop;
- dijagrame interakcije (sekvence i komunikacije) koji prikazuju:
 - rad šegrtu,
 - izračunavanje zapremine sklopa.

Rešenje:

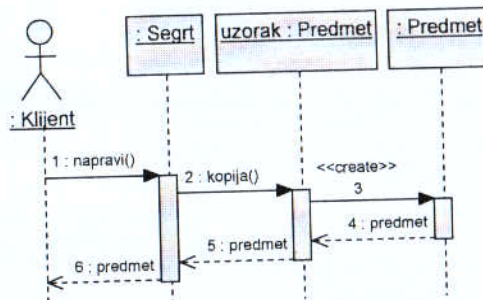
a) Ponovljeni dijagram klasa



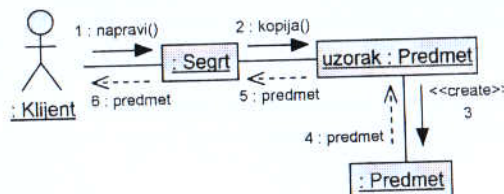
b) Dijagrami objekata



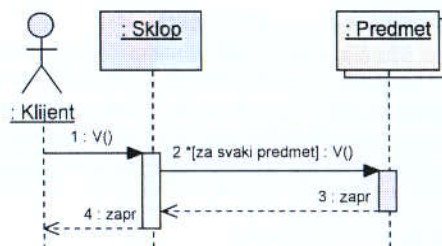
c) Dijagram sekvence rada šegrtu



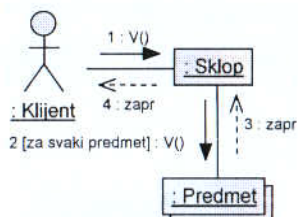
d) Dijagram komunikacije rada šegrtu



e) Dijagram sekvence računanja zapremine sklopa



f) Dijagram komunikacije računanja zapremine sklopa

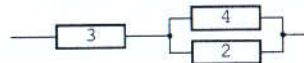


Zadatak 10 Otpornici {K1, 10.11.2006.}

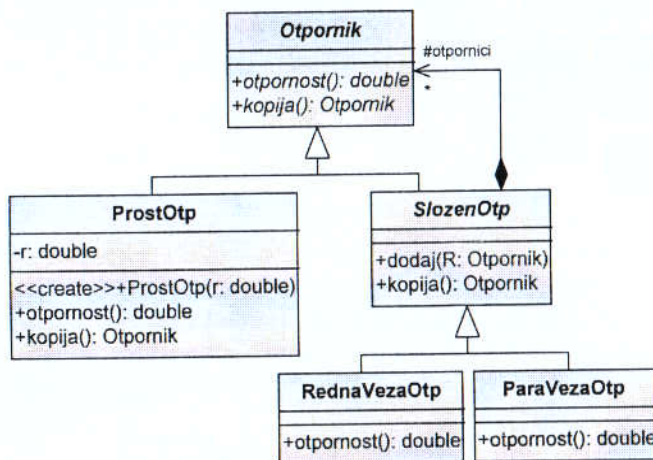
Apstraktnom otporniku može da se odredi otpornost i da se napravi njegova kopija. Prostom otporniku se zna vrednost njegove otpornosti. Apstraktan složen otpornik je otpornik koji se sastoji od proizvoljnog broja otpornika. Stvara se prazan, posle čega mogu da mu se dodaju komponentni otpornici. Redna veza otpornika je složen otpornik čija je otpornost jednaka zbiru otpornosti sadržanih otpornika. Paralelna veza otpornika je složen otpornik čija je otpornost jednaka recipročnoj vrednosti zbira recipročnih vrednosti sadržanih otpornika.

Projektovati na jeziku *UML* prethodni sistem. Priložiti:

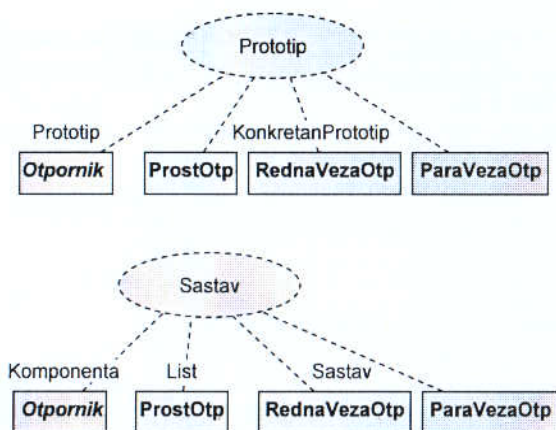
- dijagram klasa,
- prikaz korišćenih projektnih uzoraka,
- dijagram objekata koji predstavlja vezu otpornika sa slike,
- dijagram sekvence za određivanje otpornosti složenog otpornika.

**Rešenje:**

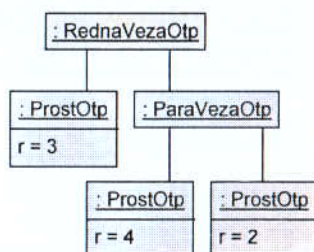
a) Dijagram klasa i projektni uzorci



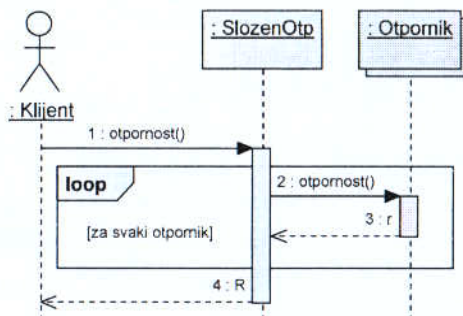
b) Projektni uzorci



c) Dijagram objekata



d) Dijagram sekvence računanja otpornosti



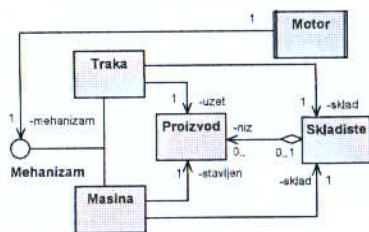
Zadatak 11 Proizvod, skladište, mehanizmi i motor

Za model proizvoda, mehanizama, skladišta i motora iz zadatka 4 nacrtati na jeziku *UML*:

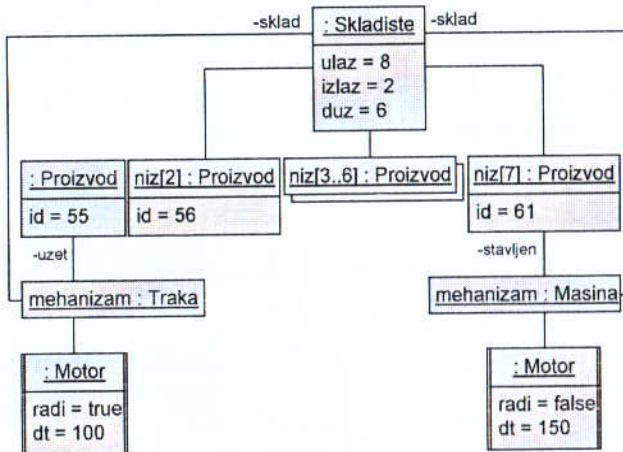
- dijagram objekata koji prikazuje skladište s nekoliko proizvoda, jedan motor koji pokreće mašinu i jedan motor koji pokreće traku;
- dijagrame interakcije (sekvence i komunikacije) koji prikazuju:
 - stvaranje mašine i izvršavanje radnje mašine,
 - stvaranje motora koji pokreće traku i rad motora.

Rešenje:

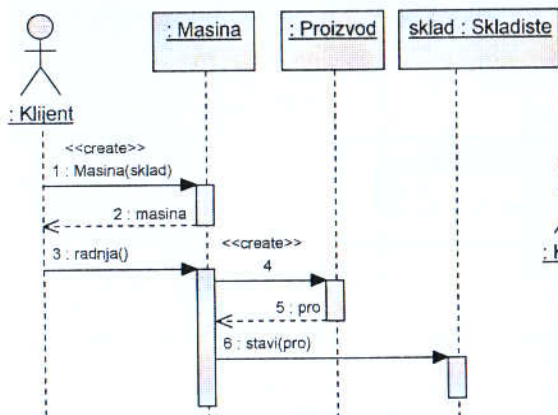
a) Ponovljeni dijagram klasa



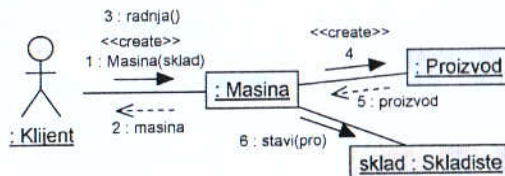
b) Dijagram objekata



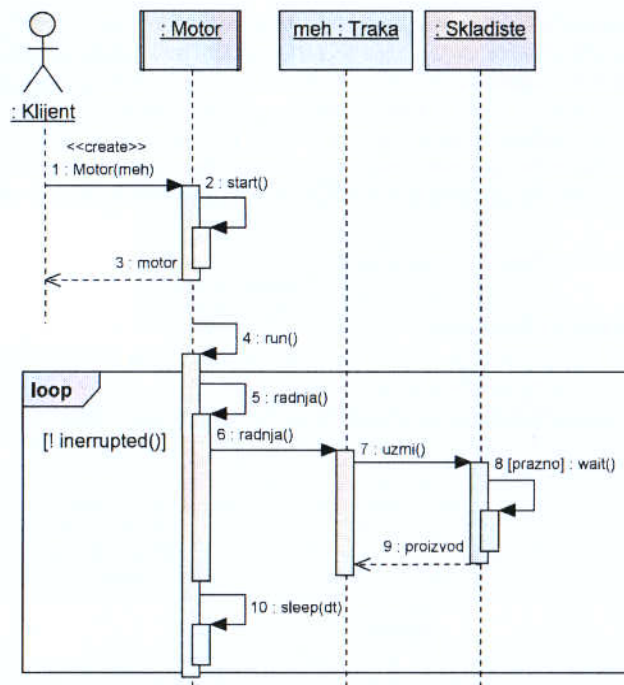
c) Dijagram sekvence korišćenja mašine



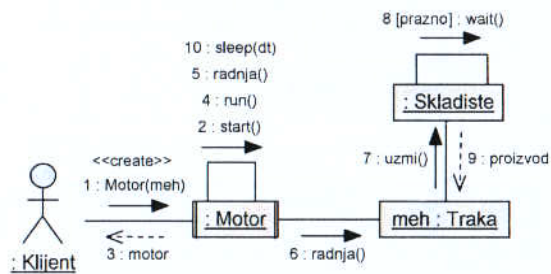
d) Dijagram komunikacije korišćenja mašine



e) Dijagram sekvence korišćenja motora



f) Dijagram komunikacije korišćenja motora



Zadatak 12 Vektor, predmeti, orijentisani predmeti, struktura i radnik {K1, 01.11.2007.}

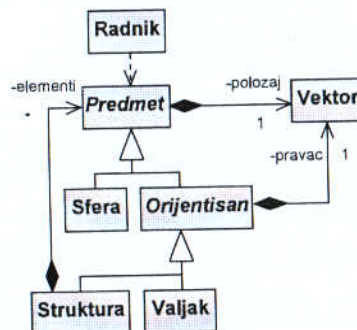
Vektor u prostoru zadaje se realnim koordinatama koje mogu da se dohvate. Vektoru može da se doda drugi vektor. Položaj apstraktnog predmeta zadaje se vektorom koji može da se dohvati. Može da se izračuna zapremina predmeta. Orijentisan predmet je predmet koji ima i pravac u prostoru određen drugim vektorom koji može da se dohvati. Sfera je predmet zadatog poluprečnika koji može da se dohvati. Valjak je orijentisan predmet zadat poluprečnikom osnove i visinom koji mogu da se dohvate. Struktura je orijentisan predmet koji može da sadrži proizvoljan broj predmeta. Stvara se prazna posle čega može da joj se doda po jedan predmet. Radnik ima ime koje može da se dohvati. Radnik može da napravi jedan predmet po svom nahodjenju.

Projektovati na jeziku *UML* prethodni sistem. Priložiti:

- dijagram klasa,
- prikaz korišćenih projektnih uzoraka,
- dijagram objekata koji predstavlja radnika koji pravi strukturu koja već sadrži jednu sferu i jedan valjak i upravo želi da strukturi doda još jednu sferu,
- dijagram sekvence za izradu jedne strukture koja sadrži samo sfere i valjke.

Rešenje:

a) Dijagram klasa



Vektor
-x: double -y: double -z: double
<<create>>+Vektor(a: double, b: double, c: double) +dohvX(): double +dohvY(): double +dohvZ(): double +dodaj(vekt: Vektor): Vektor

Predmet
<pre><<create>>#Predmet(polozej: Vektor) +dohvPolozej(): Vektor +zapremina(): double</pre>

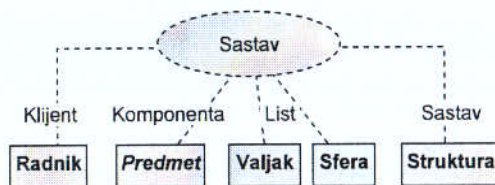
Orijentisan
<pre><<create>>#Orijentisan(polozej: Vektor, pravac: Vektor) +dohvPravac(): Vektor</pre>

Sfera
-r: double
<pre><<create>>+Sfera(polozej: Vektor, r: double) +dohvR(): double +zapremina(): double</pre>

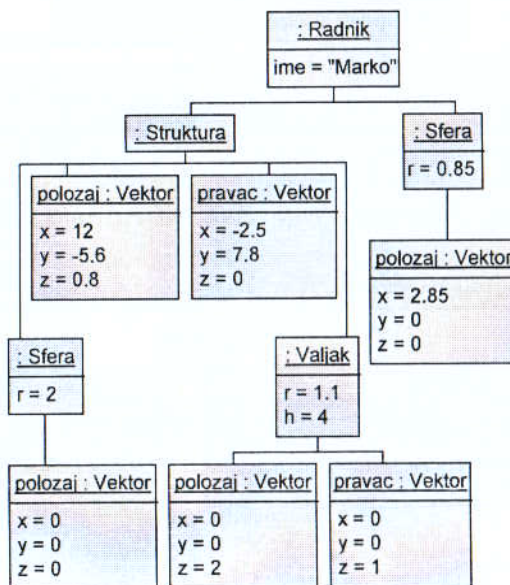
Valjak
-r: double -h: double
<pre><<create>>+Valjak(polozej: Vektor, pravac: Vektor, r: double, h: double) +dohvR(): double +dohvH(): double +zapremina(): double</pre>

Struktura
<pre><<create>>+Struktura(polozej: Vektor, pravac: Vektor) +dodaj(predmet: Predmet): Struktura +zapremina(): double</pre>

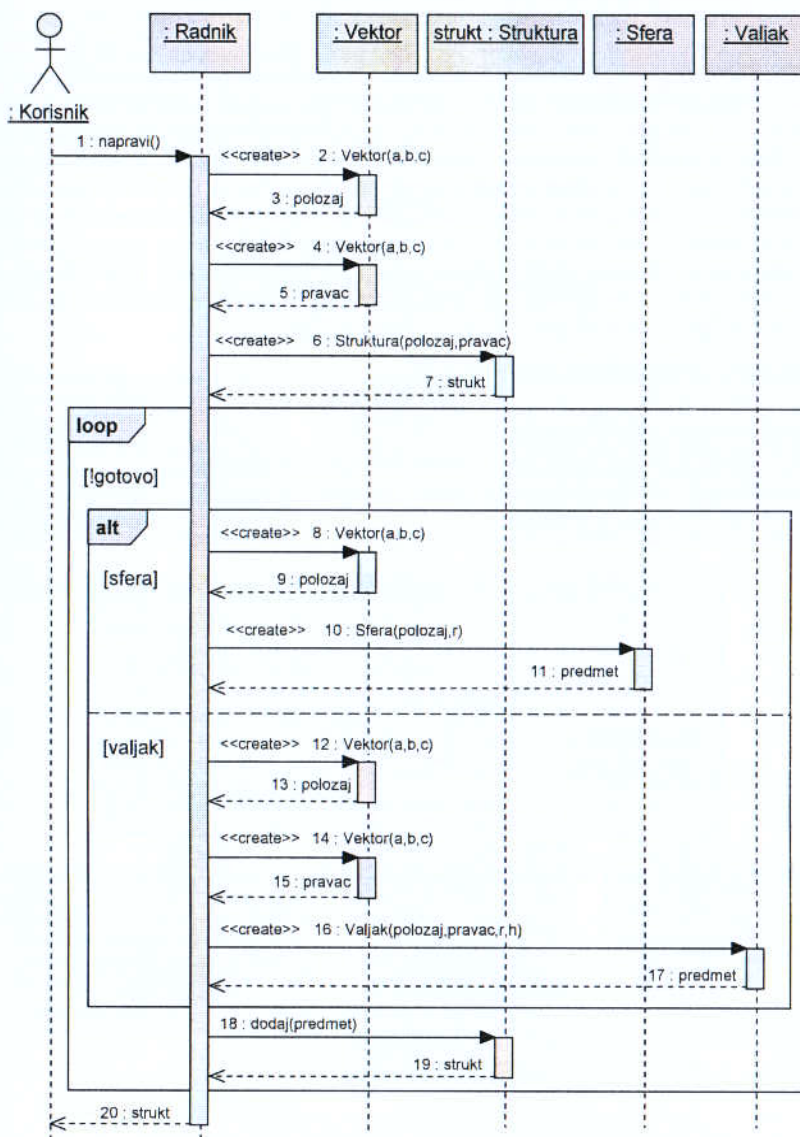
Radnik
-ime: String
<pre><<create>>+Radnik(ime: String) +dohvIme(): String +napravi(): Predmet</pre>

b) Projektni uzorak *Sastav*

c) Dijagram objekata



d) Dijagram sekvence



Zadatak 13 Simbol, font, vektor, figure, crtež, platno i aplikacija {K1, 30.10.2008.}

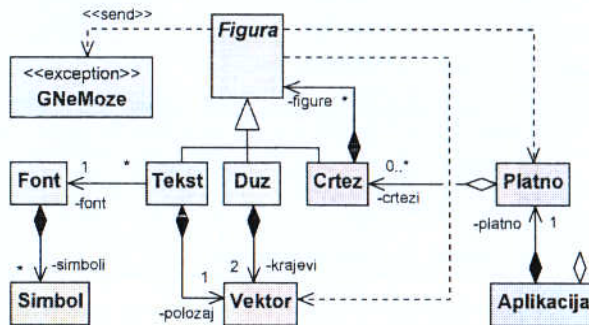
Simbol sadrži jedan znak i odnos širine i visine pri ispisivanju znaka. Može da se dohvati sadržani znak i da se odredi širina ispisivanja za datu visinu. Font sadrži proizvoljan broj simbola. Stvara se prazan posle čega se simboli dodaju jedan po jedan. Može da se dohvati broj simbola u fontu i da se dohvati simbol koji sadrži zadati znak. Vektor u ravni zadaje se realnim komponentama u pravcu koordinatnih osa. Može da se vektoru doda drugi vektor. Apstraktna figura u ravni može da se pomeri za zadati vektor pomaka, da se napravi kopija figure i da se figura prikaže na zadatom platnu. Duž je figura koja sadrži dva vektora položaja krajnjih tačaka. Tekst je figura koja sadrži zadati niz znakova koji se iscrtava simbolima zadate visine, primenom zadatog fonta sa zadatim vektorom položaja donjeg levog temena prvog simbola. Visina i font mogu da se promene. Crtež je figura koja sadrži proizvoljan broj figura. Stvara se prazan posle čega se figure dodaju jedna po jedna. Aplikacija ima jedinstveno platno koje može da iscrtava sadržane crteže. Crteži mogu da mu se dodaju i uklanjaju jedan po jedan.

Projektovati na jeziku UML prethodni sistem. Priložiti:

- dijagram klasa,
- prikaz korišćenih projektnih uzoraka,
- dijagram objekata koji prikazuje crtež s jednom duž i jednim tekstom,
- dijagram sekvence za jedno prikazivanje platna u najopštijem slučaju.

Rešenje:

a) Dijagram klasa



Simbol
-znak: char
-odnos: double
<<create>>+Simbol(zn: char, odn: double)
+znak(): char
+sirina(visina: double): double

Font
+dodaj(s: Simbol)
+brojSibmola(): int
+dohvati(znak: char): Simbol

Vektor
-x: double
-y: double
<<create>>+Vektor(x: double, y: double)
+dodaj(v: Vektor): Vektor

Figura
+dodaj(f: Figura) raises GNeMoze
+pomeri(pomak: Vektor): Figura
+kopija(): Figura
+prikazi(p: Platno)

<<exception>>
GNeMoze

Duz
<<create>>+Duz(a: Vektor, b: Vektor)
+pomeri(pomak: Vektor): Duz
+kopija(): Duz
+prikazi(p: Platno)

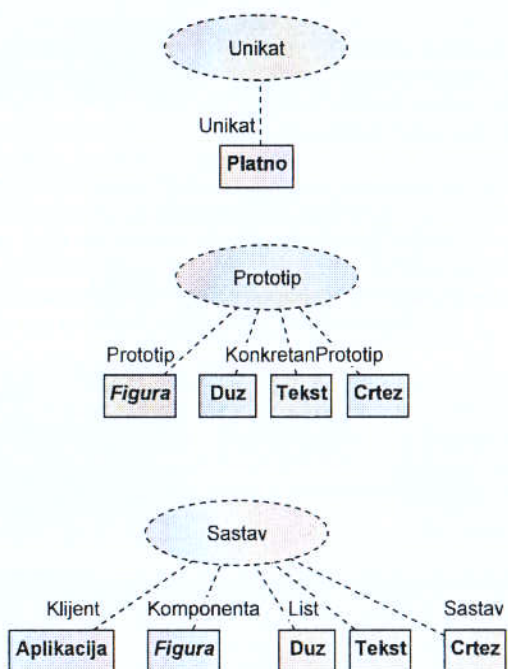
Tekst
+tekst: String
+visna: double
<<create>>+Tekst(tekst: String, f: Font, visina: double, polozaj: Vektor)
+pomeri(pomak: Vektor): Tekst
+kopija(): Tekst
+prikazi(p: Platno)
+postaviVisinu(visina: double)
+postaviFont(f: Font)

Crtez
+dodaj(f: Figura)
+pomeri(pomak: Vektor): Crtez
+kopija(): Crtez
+prikazi(p: Platno)

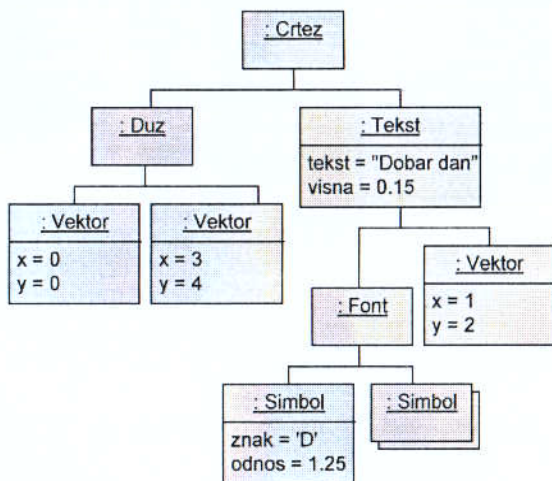
Platno
-primerak: Platno = null
<<create>>-Platno()
+dohvatiPlatno(): Platno
+dodaj(c: Crtez)
+ukloni(c: Crtez)
+prikazi()

Aplikacija
+main(varg: String[])

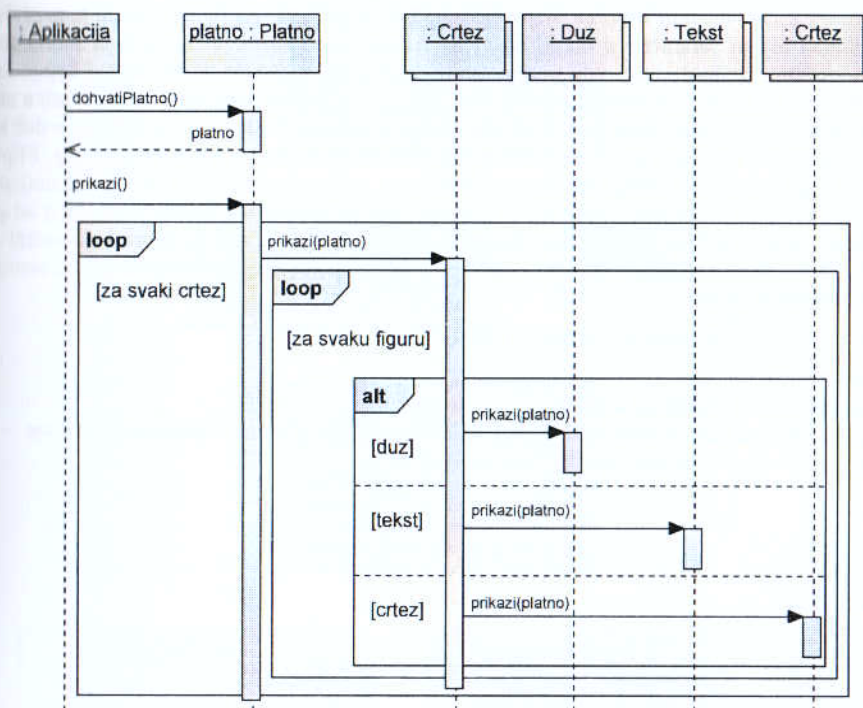
b) Projektni uzorci



c) Dijagram objekata



d) Dijagram sekvence prikazivanja platna



Zadatak 14 Tačka, figure, krive, familija krivih i crtež {K1, 30.10.2009.}

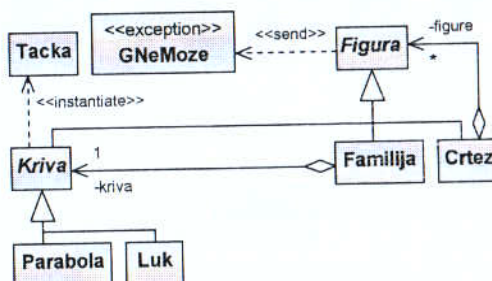
Tačka u ravni se zadaje realnim koordinatama koje mogu da se dohvate. Za figuru u ravni je predviđeno crtanje varirajući realan parametar p od p_1 do p_2 sa korakom Δp . Kriva je figura koja sadrži realan parametar q , početne vrednosti nula, koji može kasnije da se promeni. Može da se napravi kopija krive, da se napravi tačka na krivoj na osnovu zadatog parametra p i sadržanog parametra q i da se kriva nacrti spajajući pravolinijskim segmentima tačke koje odgovaraju pojedinim vrednostima parametra p duž krive (za fiksno q). Parabola je kriva koja koordinate tačaka računa formulama $x=p$ i $y=a \cdot p^2 + b \cdot q$. Eliptični luk je kriva koja koordinate tačaka računa po formulama $x=(a+q) \cdot \cos(c \cdot p + d)$ i $y=(b+q) \cdot \sin(c \cdot p + d)$. Familija krivih je figura koja sadrži zadatu krivu koju crta više puta varirajući realan parametar q od q_1 do q_2 sa korakom Δq (q_1 , q_2 i Δq zadaju se prilikom stvaranja familije). Crtež je figura koja sadrži proizvoljan broj figura. Stvara se prazan posle čega se figure dodaju pojedinačno. Crtanje crteža se sastoji od crtanja svih sadržanih figura.

Projektovati na jeziku UML prethodni sistem. Priložiti:

- dijagram klasa,
- prikaz korišćenih projektnih uzoraka,
- dijagram objekata koji prikazuje crtež sa po jednom parabolom, lukom i familijom parabola;
- dijagram sekvence za crtanje crteža u najopštijem slučaju.

Rešenje:

a) Dijagram klasa



Tacka
-x: float -y: float
<<create>>+Tacka(x: float, y: float) +dohvX(): double +dohvY(): double

Figura
+crtaj(p1: float, p2: float, dp: float) +dodaj(f: Figura) raises GNeMoze

<<exception>> GNeMoze

Kriva
#q: float = 0
+postaviQ(q: float) +kopija(): Kriva +tacka(p: float): Tacka +crtaj(p1: float, p2: float, dp: float)

Parabola
-a: float -b: float
<<create>>+Parabola(a: float, b: float) +kopija(): Parabola +tacka(p: float): Tacka

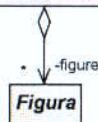
Luk
-a: double -b: double -c: double -d: double
<<create>>+Luk(a: float, b: float, c: float, d: float) +kopija(): Luk +tacka(p: float): Tacka

Familija
-q1: float -q2: float -dq: float
<<create>>+Familija(k: Kriva, q1: float, q2: float, dq: float) +crtaj(p1: float, p2: float, dp: float)



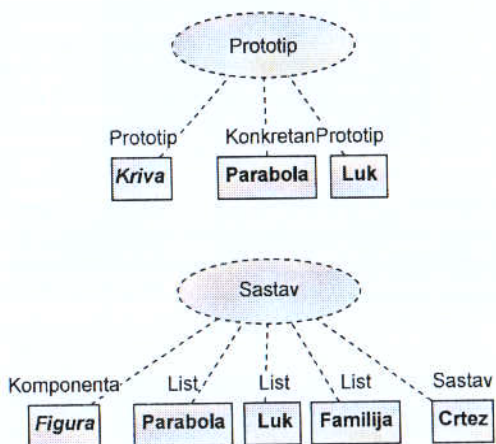
Krivu crta
vise puta

Crtez
+crtaj(p1: float, p2: float, dp: float) +dodaj(f: Figura)

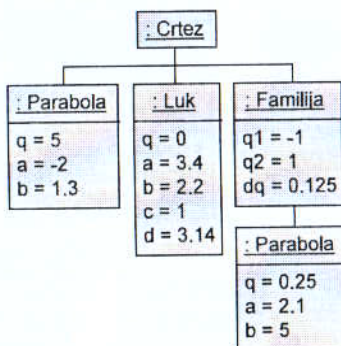


Svaku figuru
crta jednom

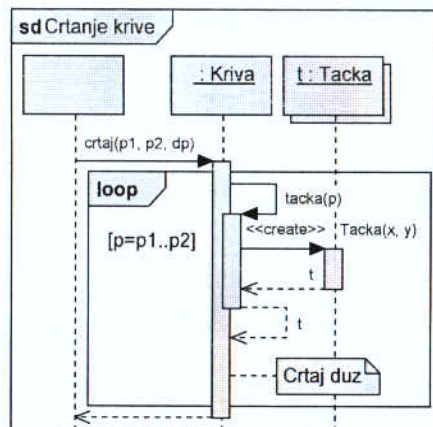
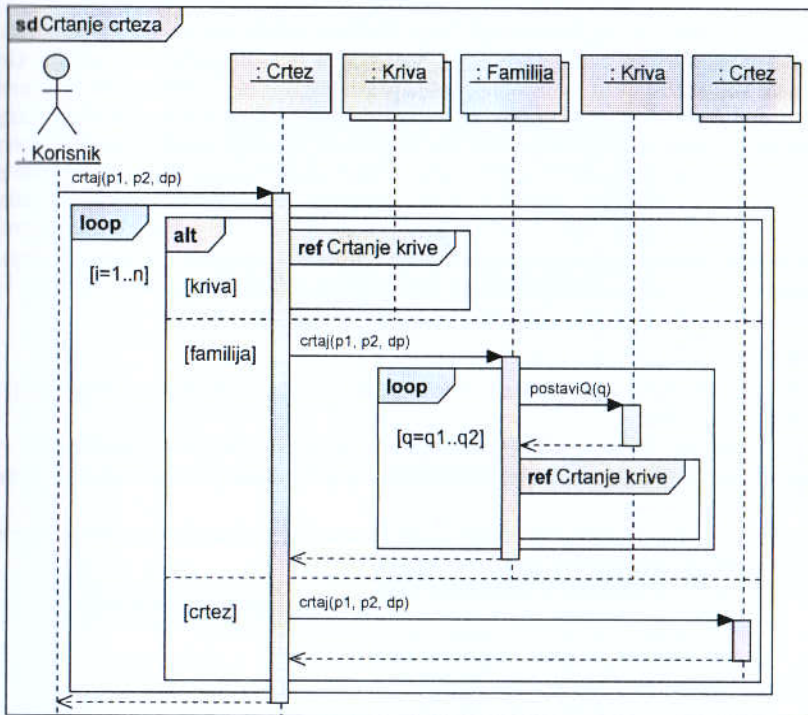
b) Projektni uzorci



c) Dijagram objekata



d) Dijagram sekvence prikazivanja crteža



Zadatak 15 Potrošači, elektrane, elektrodistribucija {K1, 1.11.2010.}

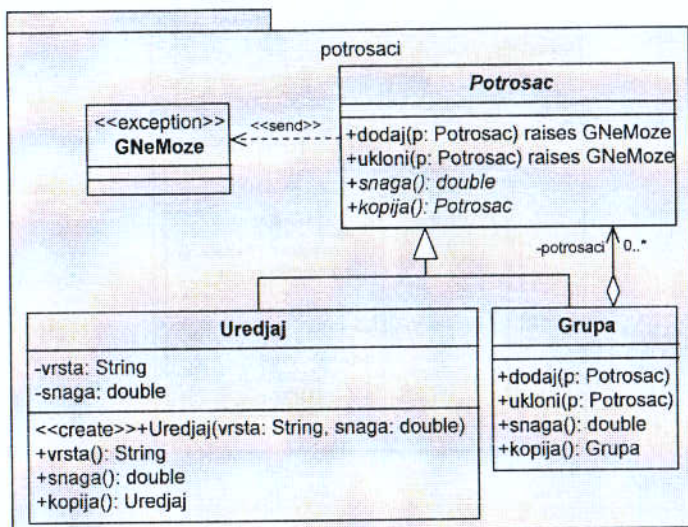
Potrošaču električne energije može da se odredi snaga potrošnje i da se napravi kopija. Električni uređaj je potrošač zadanog naziva vrste i snage potrošnje. Može da se dohvati naziv vrste uređaja. Grupa potrošača je potrošač koji se sastoji od proizvoljnog broja potrošača. Stvara se prazna posle čega može da se doda i ukloni zadati potrošač. Elektrana ima zadati naziv i snagu proizvodnje električne energije koji mogu da se dohvate. Termoelektrana, hidroelektrana i nuklearna elektrana su elektrane. Jedinstvena električna distribucija može biti povezana sa proizvoljnim brojem elektrana i može na nju da bude prikazan proizvoljan broj potrošača. Stvara se prazna posle čega može da se poveže sa i odveže od zadate elektrane, odnosno da se na nju prikači i od nje otkači zadati potrošač. Može da se odredi ukupna proizvodna snaga svih povezanih elektrana i ukupna potrošna snaga prikazanih potrošača. Ukupna potrošnja ne sme biti veća od ukupne proizvodnje električne snage. Povratna vrednost pri odvezivanju elektrane i prikačivanju potrošača je indikator dozvoljenosti radnje.

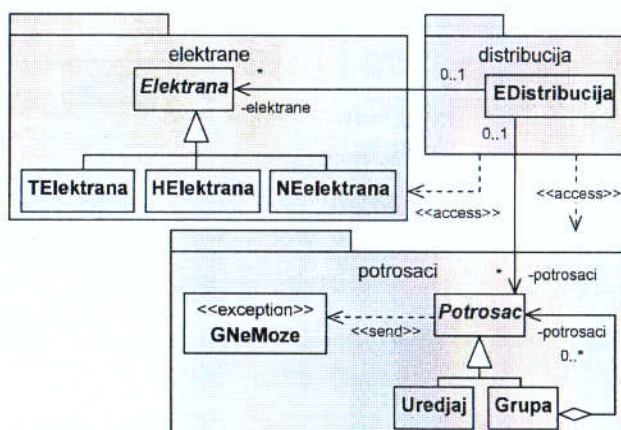
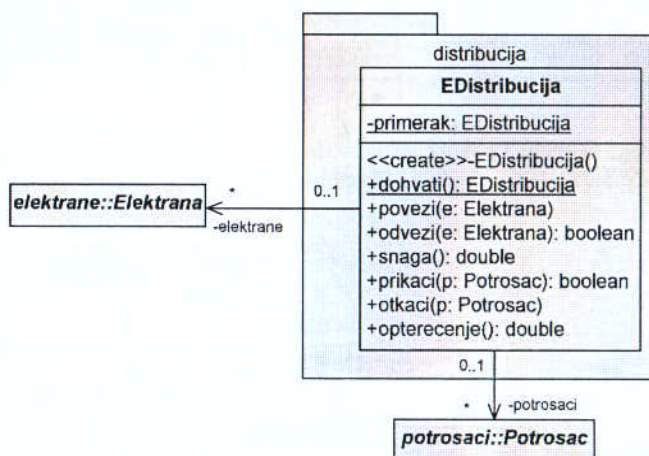
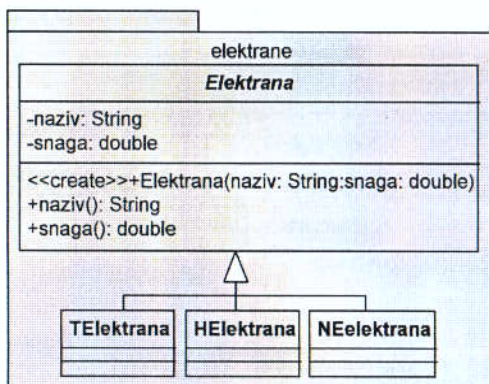
Projektovati na jeziku UML prethodni sistem. Priložiti:

- dijagram klasa (detaljan sadržaj klasa i odnose među klasama na potrebnom broju slika) logično razvrstanih po paketima;
- prikaz korišćenih projektnih uzoraka (naglasiti ako je uzorak atipičan, uz obrazloženje);
- dijagram objekata koji prikazuje elektrodistribuciju s dve elektrane, jednim uređajem i jednom grupom od dva uređaja;
- dijagram komunikacije i odgovarajući dijagram sekvence koji prikazuje stvaranje jednog uređaja i njegovo prikačivanje na distribuciju.

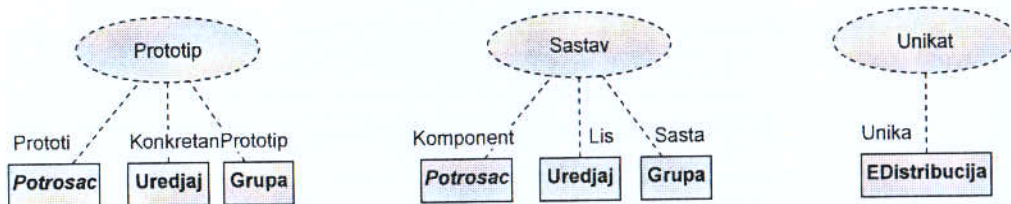
Rešenje:

a) Dijagrami klasa i paketa

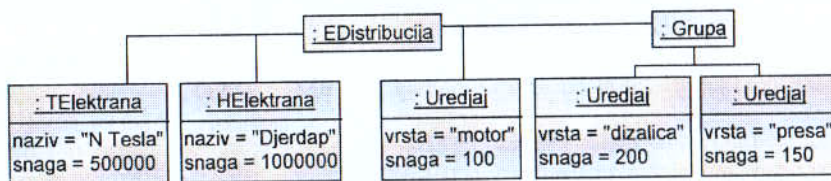




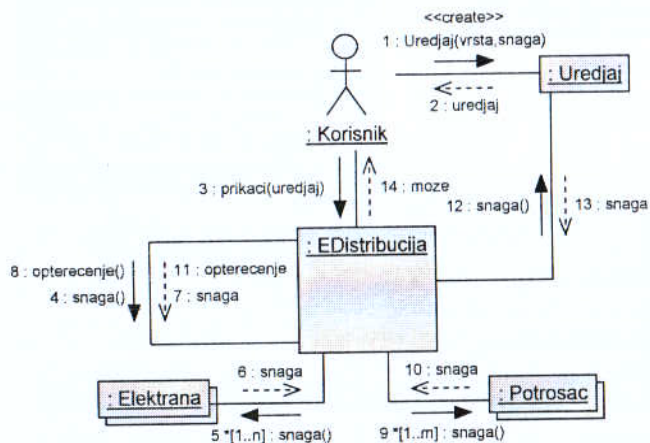
b) Projektni uzorci



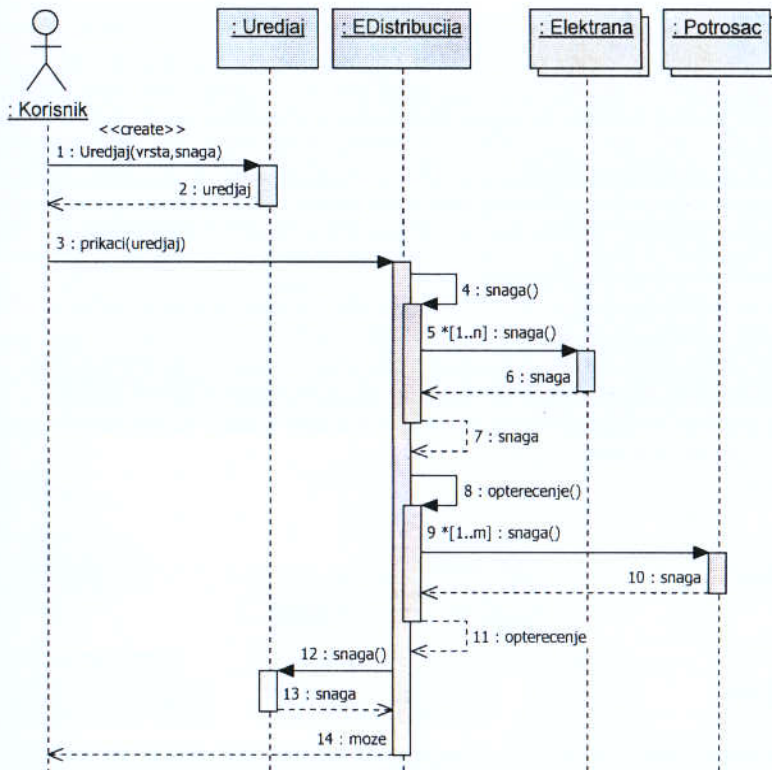
c) Dijagram objekata



d) Dijagram komunikacije stvaranja i priključivanja uređaja



e) Dijagram sekvence stvaranja i priključivanja uređaja



Zadatak 16 Lavirint {K1, 28.10.2011.}

Tačka ima x i y koordinatu koje se mogu dohvatiti. Element ima jednoslovnu oznaku orijentacije (**H** – horizontalna, **V** – vertikalna), zadatu početnu tačku i dužinu u smeru rastuće odgovarajuće koordinate. Element može da se kopira. Zid, ulaz i izlaz su elementi. Zid ima boju koja može da se dohvati. Boja ima tri celobrojne komponente. Lavirintu se mogu dohvatiti ulaz i izlaz. Prost lavirint sadrži proizvoljan broj elemenata od kojih je samo jedan ulaz i jedan izlaz. Pokušaj dodavanja drugog ulaza ili izlaza je greška. Složen lavirint sadrži lavirinte proizvoljne vrste. Ulaz složenog lavirinta je ulaz prvog sadržanog lavirinta. Izlaz složenog lavirinta je izlaz poslednjeg sadržanog lavirinta. Sadržanim lavirintima se spajaju izlaz prethodnog s ulazom narednog lavirinta. Lavirinte stvara jedinstvena fabrika lavirinata.

Projektovati na jeziku UML prethodni sistem. Priložiti:

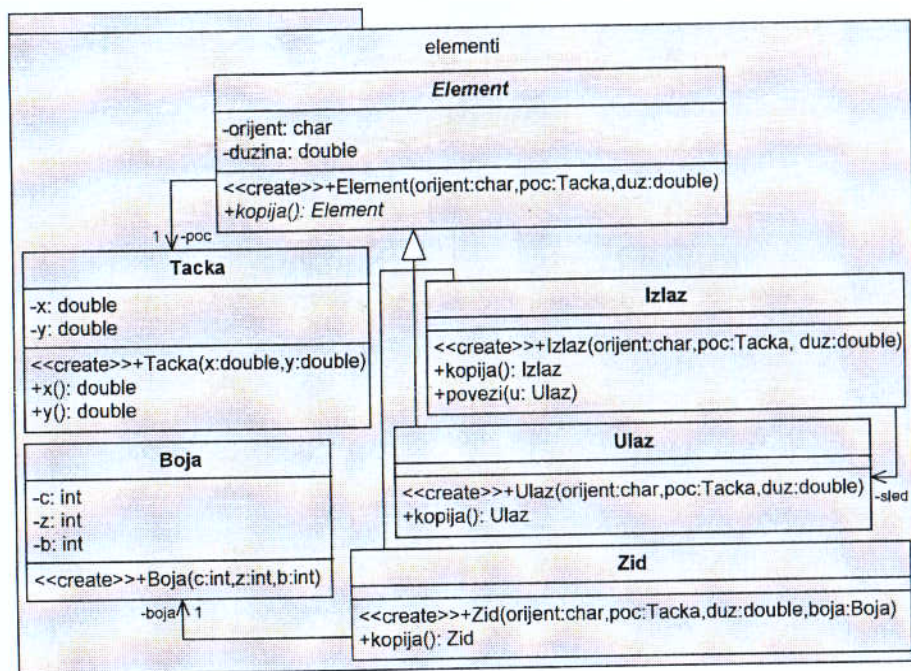
- dijagram klasa (detaljan sadržaj klasa i odnose među klasama na potrebnom broju slika) logično razvrstanih po paketima;
- prikaz korišćenih projektnih uzoraka (naglasiti ako je uzorak atipičan, uz obrazloženje);
- dijagram objekata koji prikazuje jedan složen lavirint s dva neprazna prosta lavirinta;
- dijagram sekvence i dijagram komunikacije koji prikazuje gradnju jednog složenog lavirinta koji sadrži samo proste lavirinte.

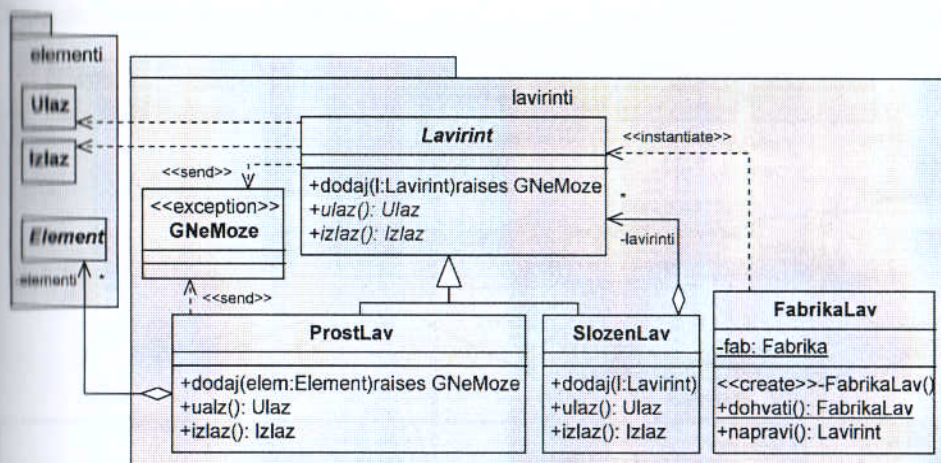
Rešenje:

a) Dijagram paketa

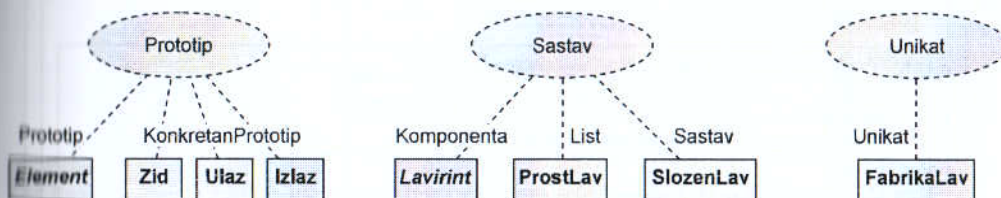


b) Dijagrami klasa

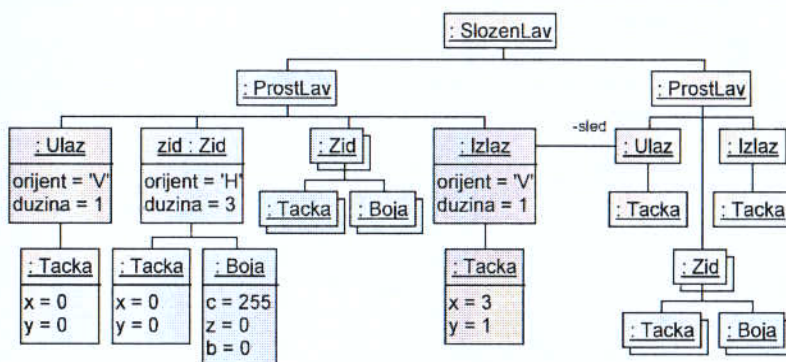




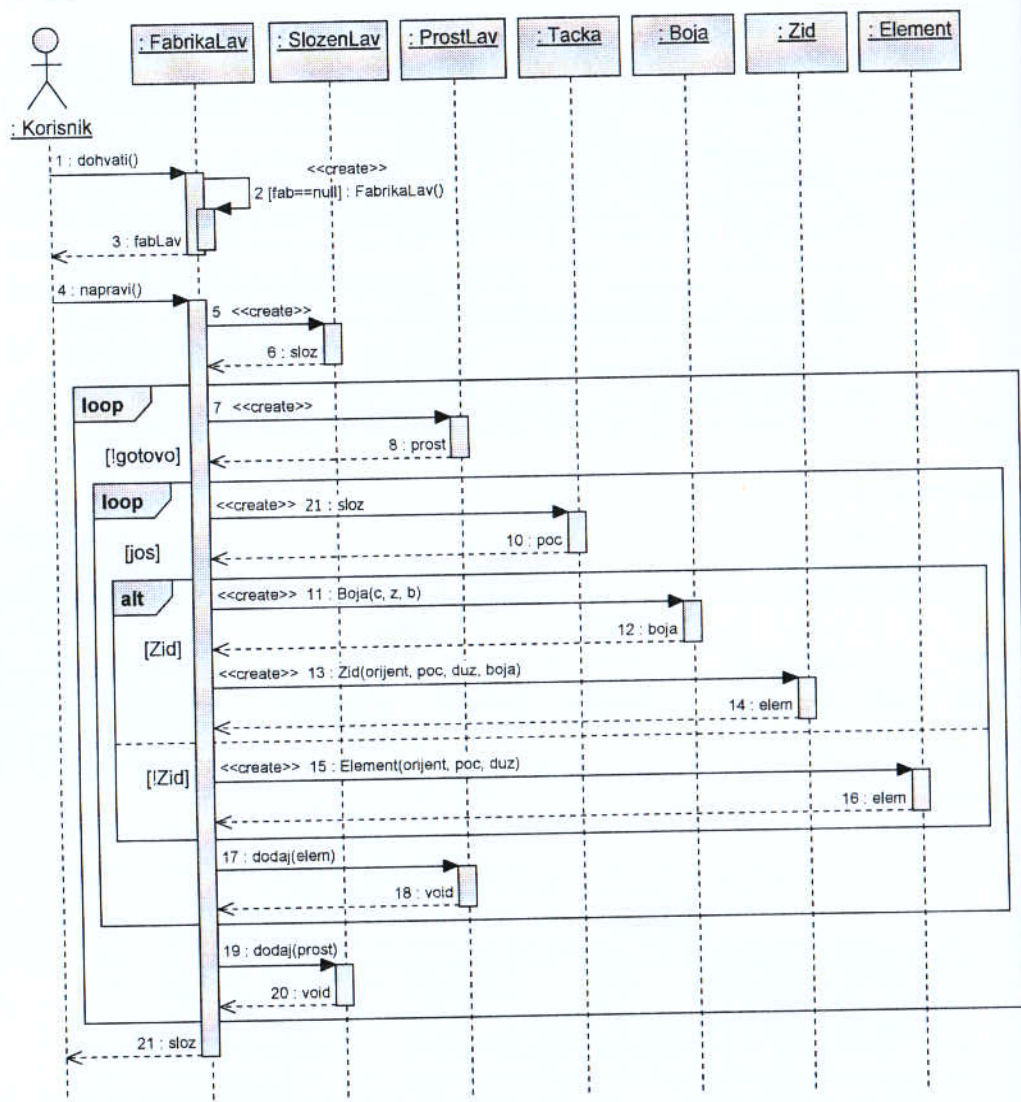
e) Projektni uzorci



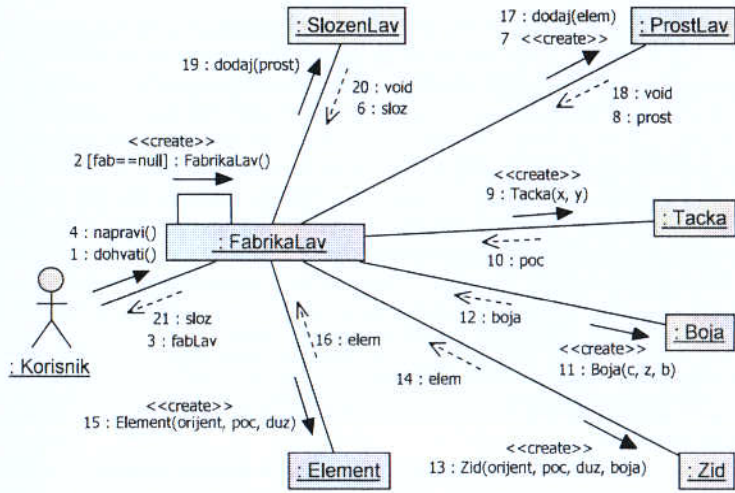
d) Dijagram objekata



e) Dijagram sekvence gradnje lavirinta



f) Dijagram komunikacije gradnje lavirinta



Zadatak 17 Lica, naučni radovi, naučne oblasti i zbornici radova {K1, 23.10.2012.}

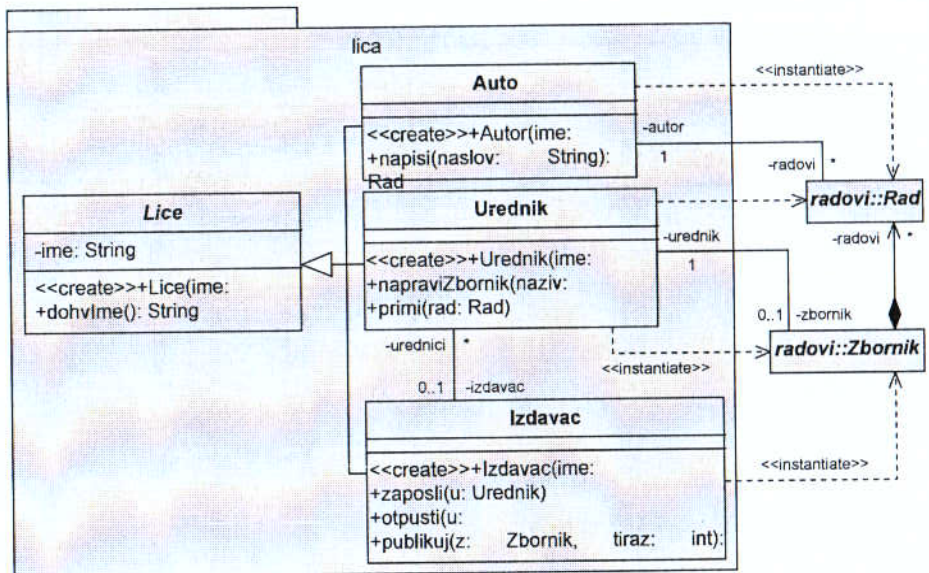
Lice ima ime koje može da se dohvatiti. Autor je lice koje može da napiše naučni rad zadatog naslova. Urednik zbornika naučnih radova je lice koje može da napravi prazan zbornik naučnih radova i da primi rad za taj zbornik. Izdavač je lice koje ima određen broj urednika koje zapošljava i otpušta pojedinačno i publikuje zadati zbornik praveći zadat broj njegovih kopija. Naučni rad ima jednoznačan identifikator (nisku), naslov i autore koji mogu da se dohvate i može da mu se napravi kopija. Rad može da pripada većem broju oblasti koje mu se dodeljuju pojedinačno. Rad može biti teorijski ili praktičan. Identifikator rada dobija se od posebnog servera koji obezbeđuje njegovu jednoznačnost. Zbornik radova ima naziv i urednika. Može da se doda proizvoljan broj radova pojedinačno i da se napravi kopija zbornika. Zbornik može biti štampan ili elektronski. Klasifikacija naučnih oblasti je hijerarhijska do proizvoljne dubine. Oblast, koja može biti prosta ili složena, ima naziv. Može da joj se doda zadata podoblast, da joj se doda zadati rad i da se pronade oblast zadatog naziva. Greška je dodavanje podoblasti ili rada ako data podoblast, odnosno rad već postoji, kao i dodavanje podoblasti prostoj oblasti.

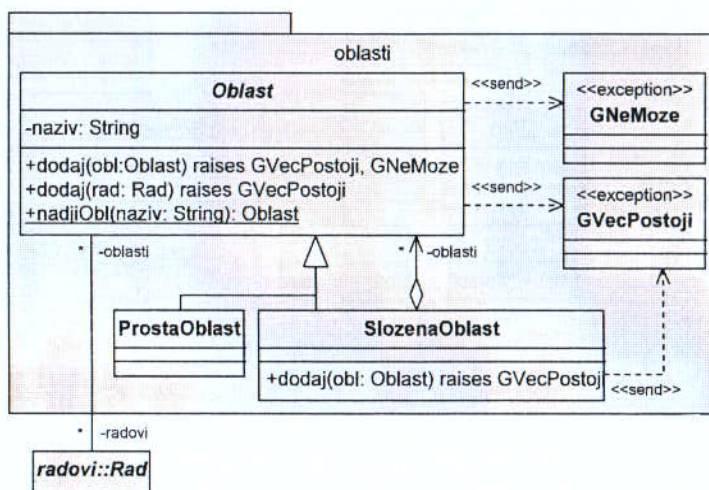
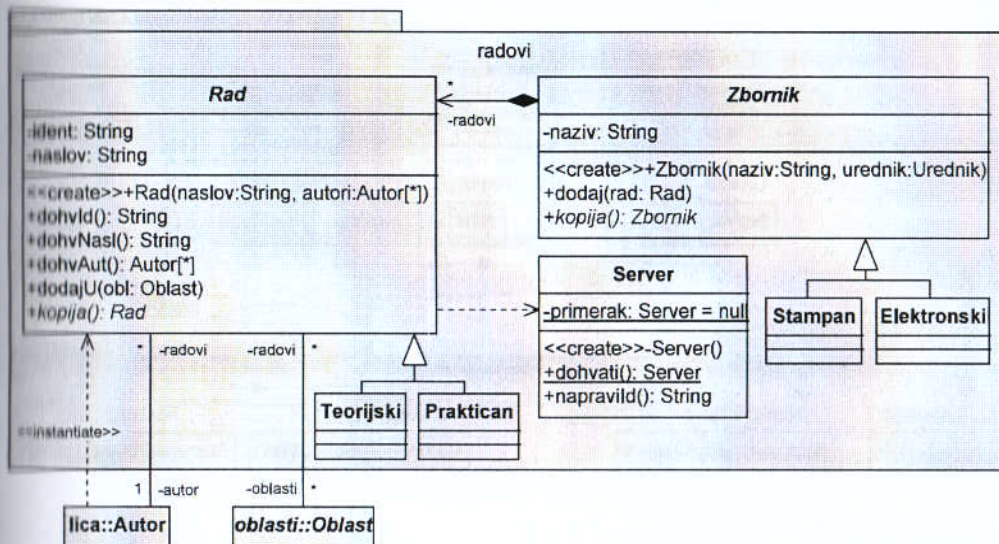
Projektovati na jeziku UML prethodni sistem. Priložiti:

- dijagram klasa (detaljan sadržaj klasa i odnose među klasama na potrebnom broju slika) logično razvrstanih po paketima;
- prikaz korišćenih projektnih uzoraka (naglasiti ako je uzorak atipičan, uz obrazloženje);
- dijagram objekata koji prikazuje zbornik sa dva rada sa po jednim autorom. od kojih svaki rad pripada po jednoj prostoj oblasti neposredno ispod korena stabla oblasti;
- dijagram komunikacije koji prikazuje nastanak rada i njegovo uključivanje u oblasti i zbornik.

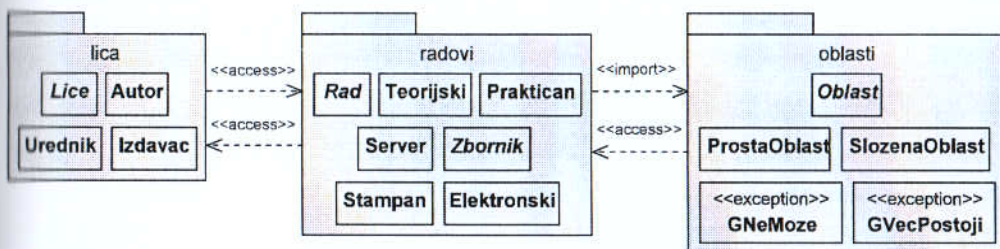
Rešenje:

a) Dijagrami klasa

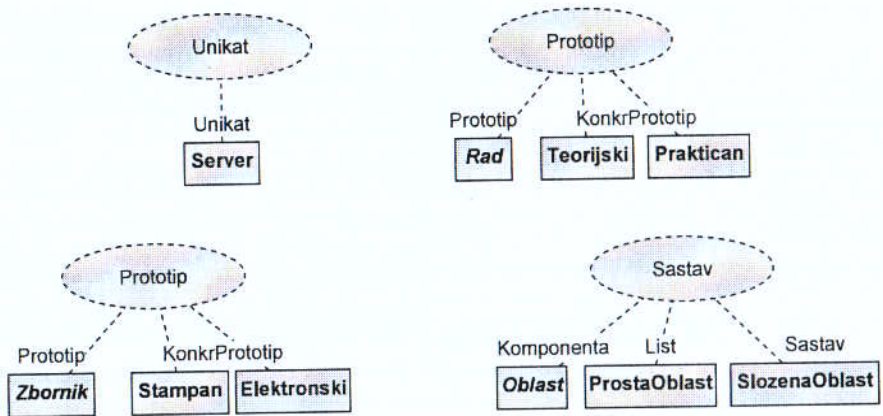




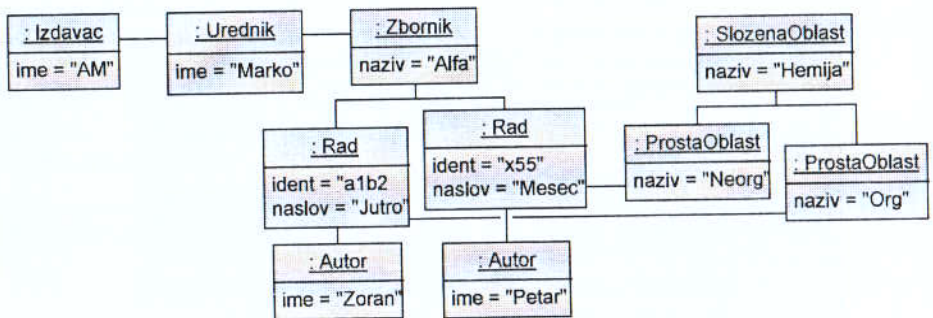
b) Dijagram paketa



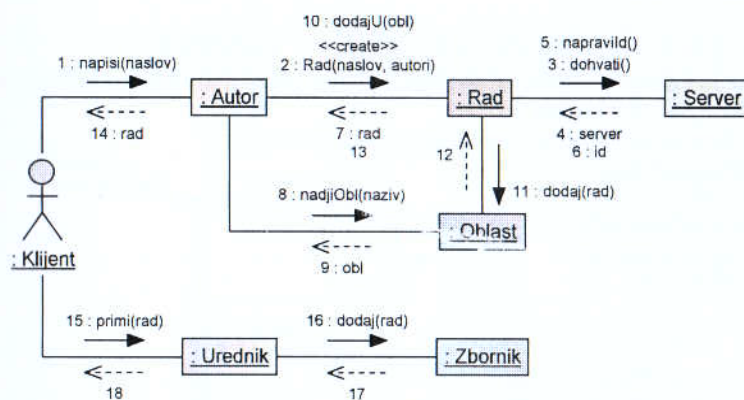
c) Projektni uzorci



d) Dijagram objekata



g) Dijagram komunikacije stvaranja rada



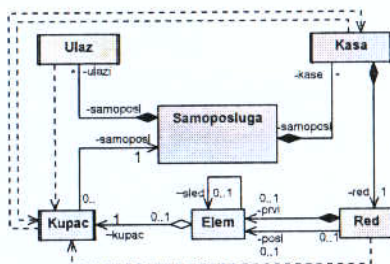
Zadatak 18 Samoposluga

Za model samoposluge iz zadatka 5 nacrtati na jeziku *UML*:

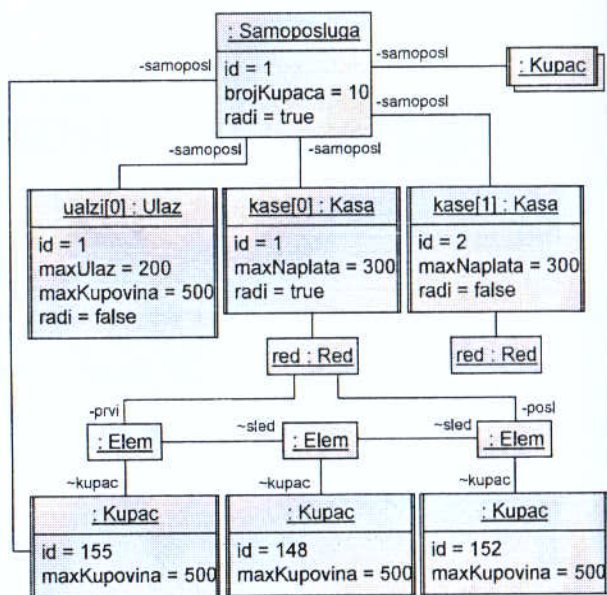
- dijagram objekata koji prikazuje samoposlugu s jednim ulazom, dve kase, nekoliko kupaca od kojih neki stoje u redu ispred jedne od kasa;
- dijagrame interakcije (sekvence i komunikacije) koji prikazuju:
 - stvaranje samoposluge,
 - otvaranje samoposluge,
 - zatvaranje samoposluge,
 - uništavanje samoposluge,
 - životni vek kupca.

Rešenje:

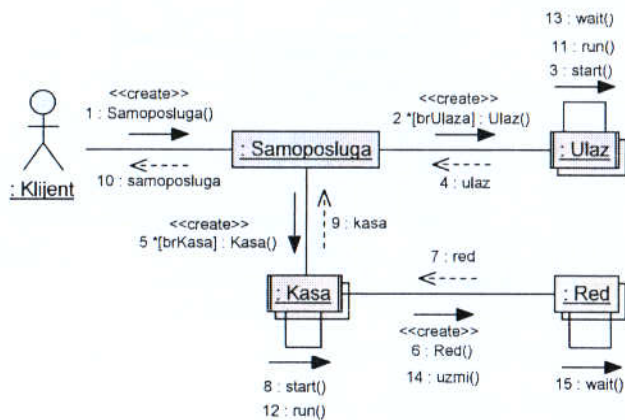
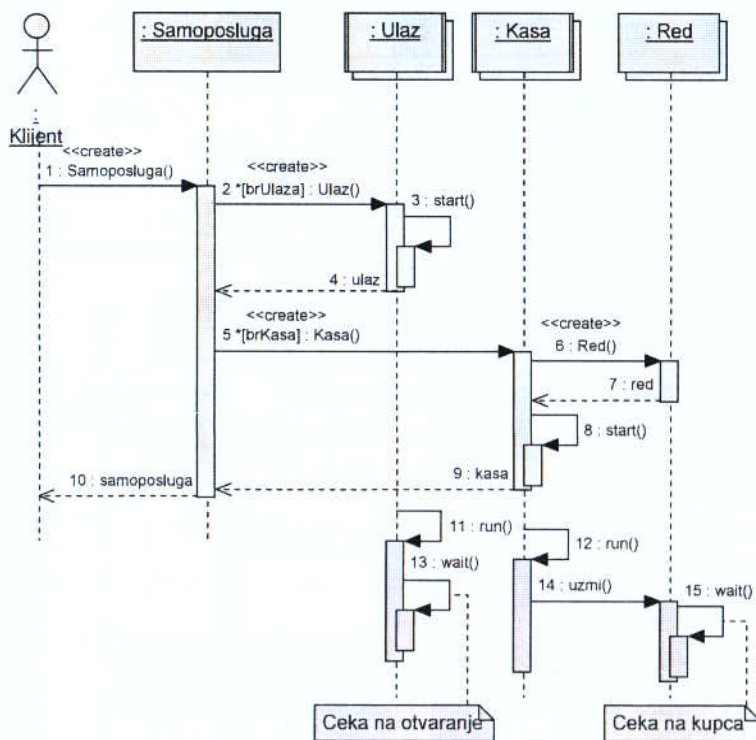
a) Ponovljeni dijagram klasa



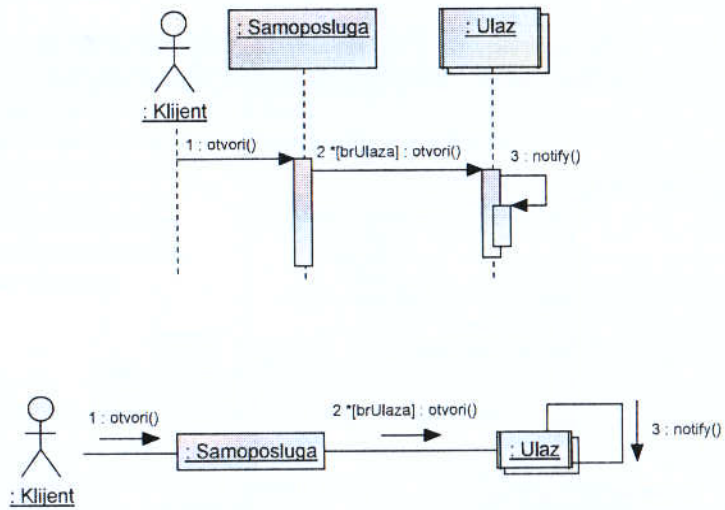
b) Dijagram objekata



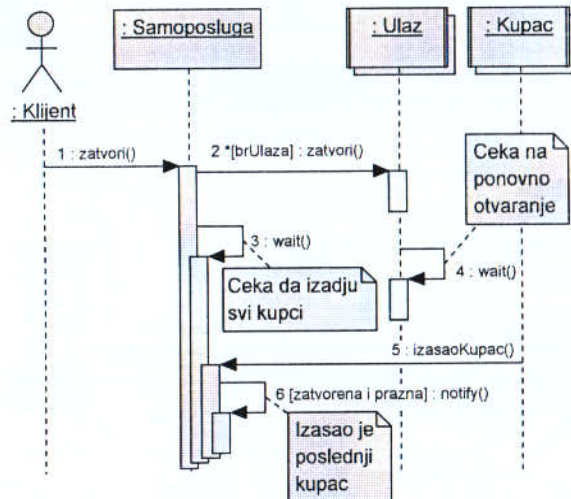
e) Dijagrami interakcije stvaranja samoposluge

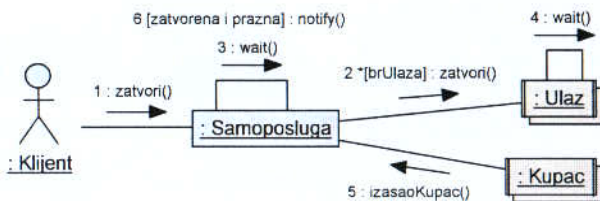


d) Dijagrami interakcije otvaranja samoposluge

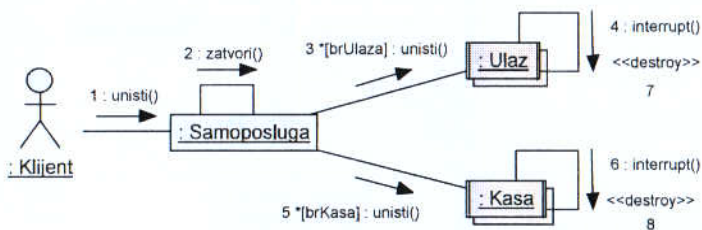
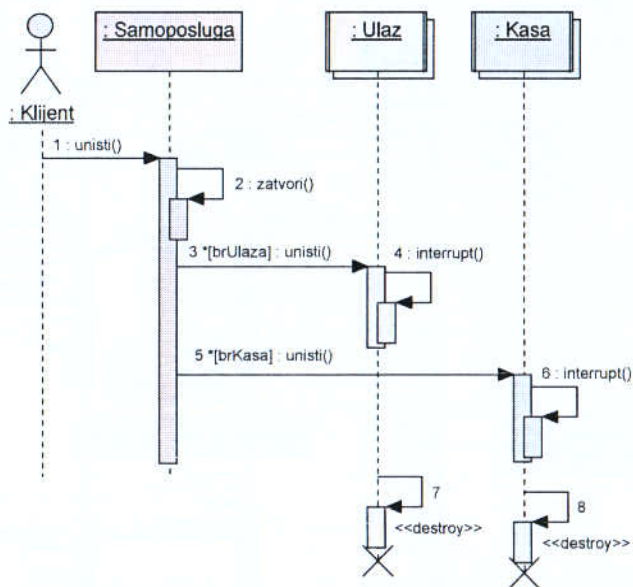


e) Dijagrami interakcije zatvaranja samoposluge

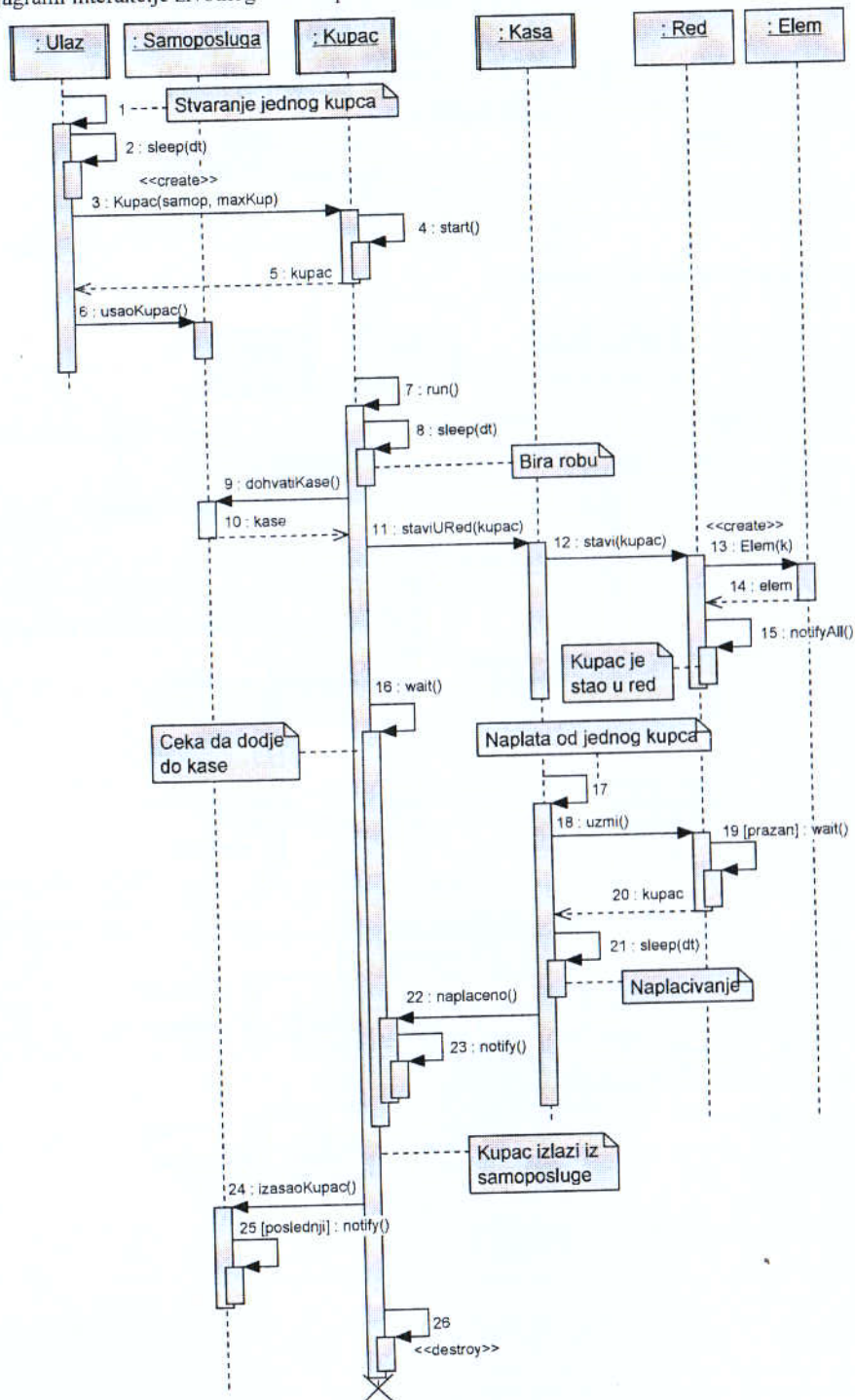


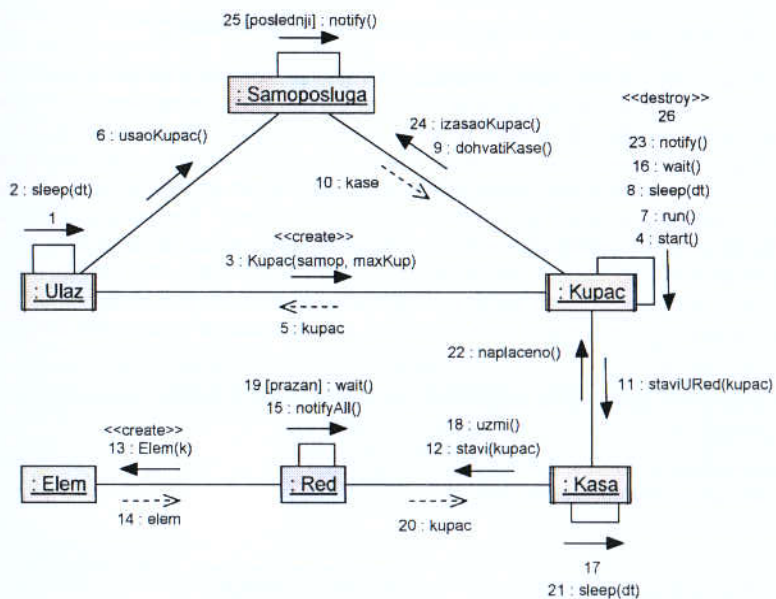


f) Dijagrami interakcije uništavanja samoposluge



g) Dijagrami interakcije životnog veka kupca





Zadatak 19 Distributeri, klijenti, pošiljke i raspodele (projektni uzorak *Posmatrač*)

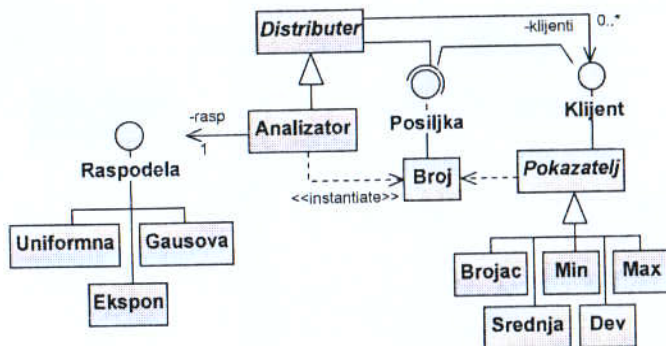
Apstraktan distributer može da isporučuje apstraktnu pošiljku svakom apstraktnom klijentu koji mu se prijavljuje radi prijema pošiljki. Klijenti mogu da se odjavljuju ako više nisu zainteresovani za prijem pošiljki. Apstraktan generator slučajnih brojeva na svaki zahtev daje jedan pseudoslučajan realan broj po nekoj raspodeli. Moguće raspodele su: uniformna, Gausova i eksponencijalna. Analizator je distributer koji na zahtev generiše zadati broj pseudoslučajnih brojeva. Te brojeve isporučuje, jedan po jedan, svim apstraktnim statističkim pokazateljima koji predstavljaju njegove klijente. Raspodela generisanih brojeva može da se postavlja po želji. Apstraktan statistički pokazatelj je klijent koji obrazuje neki statistički pokazatelj primljenih brojeva u pošiljkama. Može da se dovodi u početno stanje radi početka nove analize i da se dohvati rezultat analize primljenih brojeva do momenta zahteva. Mogući statistički pokazatelji su: broj uzoraka, najmanja i najveća vrednost među uzorcima, srednja vrednost i standardna devijacija uzoraka.

Projektovati na jeziku *UML* i napisati na jeziku *Java* prethodni sistem. Priložiti:

- dijagrame klasa,
- prikaz korišćenih projektnih uzoraka,
- dijagram sekvence rada analizatora,
- programsku realizaciju.

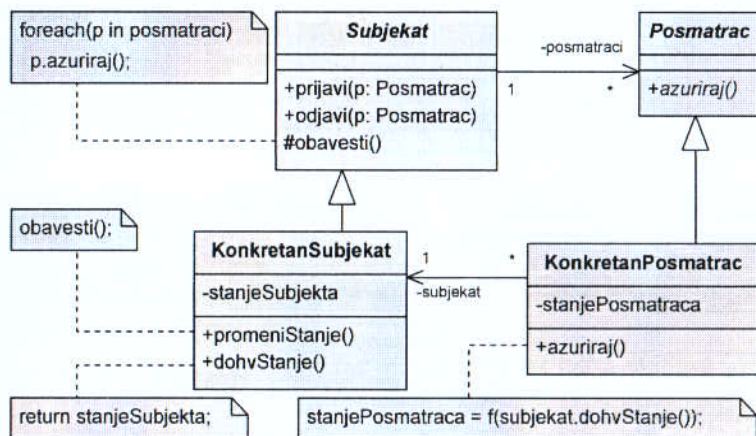
Rešenje:

a) Dijagram klasa

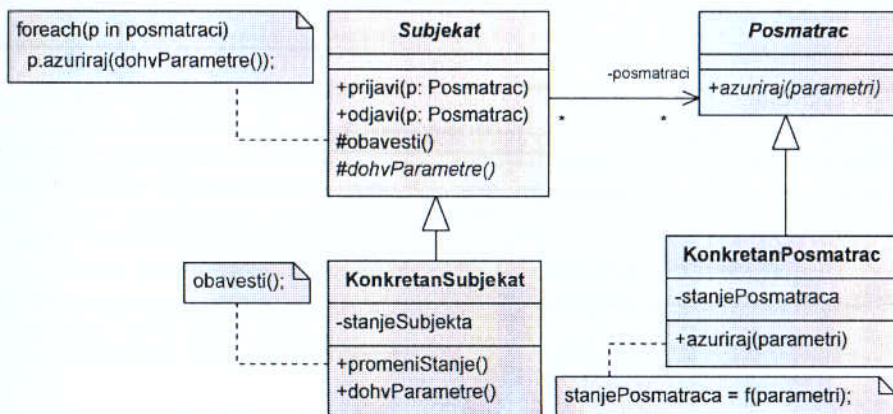


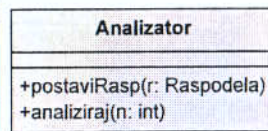
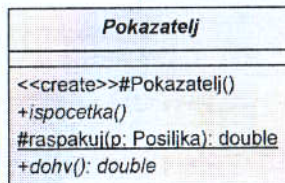
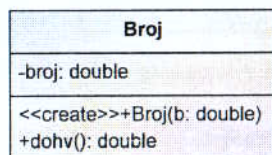
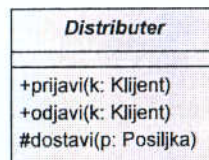
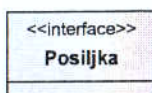
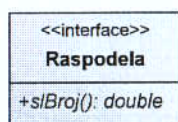
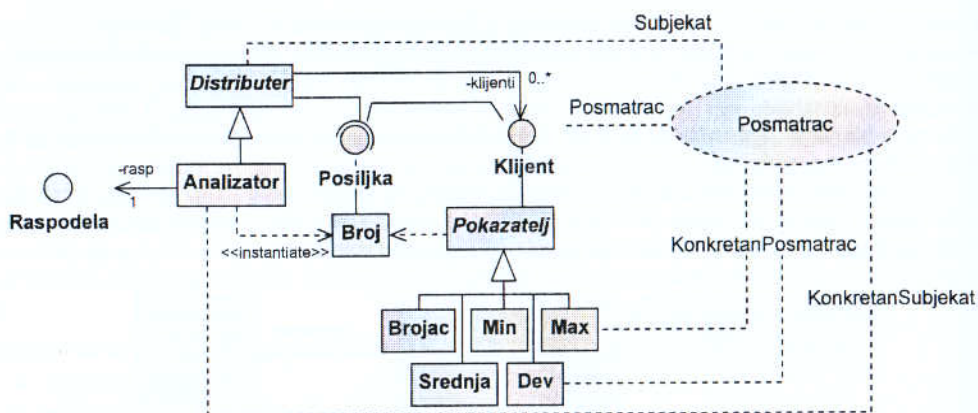
b) Projektni uzorak **Posmatrač (Observer)**

- objektni uzorak ponašanja
- distribucija informacija većem broju nezavisnih zainteresovanih (registrovanih) klijenata
- subjekat obaveštava svoje posmatrače o promeni svog stanja da bi i posmatrači ažurirali svoja stanja
- model **preuzimanja**:
 - subjekat samo signalizira da je došlo do promene stanja, posle čega posmatrači treba da traže podatke o toj promeni
 - mana: posmatrači treba da znaju za detalje subjekta da bi znali kako i koje podatke mogu da traže
 - prednost: lako uvođenje novih vrsta posmatrača sa novim potrebama



- model **isporučivanja**:
 - subjekat dostavlja podatke o promeni svog stanja
 - prednost: posmatrači ne treba da znaju ništa o detaljima subjekta
 - mana: subjekat treba da zna za potrebe svojih posmatrača i teško se prilagođava novim vrstama posmatrača sa novim potrebama



c) Primer uzorka *Posmatrač* (model isporučivanja)

```
// Raspodela.java
public interface Raspodela {
    public double slBroj();
}
```

```
// Posiljka.java
public interface Posiljka {}
```

```
// Klijent.java
public interface Klijent {
    public void primi(Posiljka p);
}
```



```
// Distributer.java
public abstract class Distributer {
    private class Elem {
        Kljent kljent; Elem sled;
        Elem(Kljent k) {
            kljent = k; sled = null;
            if (prvi == null) prvi = this; else posl.sled = this;
            posl = this;
        }
    }

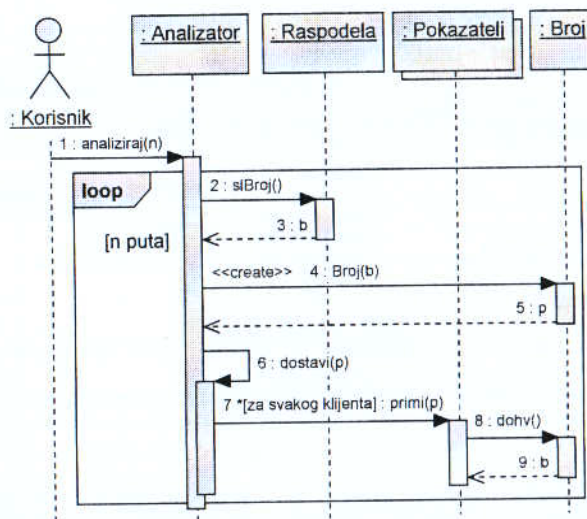
    private Elem prvi, posl;
    public void prijavi(Kljent k) { new Elem(k); }
    public void odjavi(Kljent k) {
        Elem tek = prvi, pret = null;
        while (tek!=null && tek.kljent!=k) { pret = tek; tek = tek.sled; }
        if (tek != null) {
            if (pret == null) prvi = tek.sled; else pret.sled = tek.sled;
            if (tek == posl) posl = pret;
        }
    }
    protected void dostavi(Posiljka p) {
        for (Elem tek=prvi; tek!=null; tek=tek.sled) tek.kljent.primi(p);
    }
}
```

```
// Broj.java
public class Broj implements Posiljka {
    private double broj;
    public Broj(double b) { broj = b; }
    public double dohv() { return broj; }
}
```

```
// Pokazatelj.java
public abstract class Pokazatelj implements Kljent {
    protected Pokazatelj() { ispocetka(); }
    public abstract void ispocetka();
    protected static double raspakuj(Posiljka p)
    { return ((Broj)p).dohv(); }
    public abstract double dohv();
}
```

```
// Analizator.java
public class Analizator extends Distributer {
    private Raspodela rasp;
    public void postaviRasp(Raspodela r) { rasp = r; }
    public void analiziraj(int n) {
        for (int i=0; i<n; i++)
            dostavi(new Broj(rasp.slBroj(i)));
    }
}
```


d) Dijagram sekvence rada analizatora



e) Ostale klase

Uniformna
-min: double -max: double
<<create>>+Uniformna(min: double, max: double) +slBroj(): double

Gausova
-sr: double -dev: double
<<create>>+Gausova(sr: double, dev: double) +slBroj(): double

Ekspon
-sr: double
<<create>>+Ekspon (sr: double) +slBroj(): double

```
// Uniformna.java
public class Uniformna implements Raspodela {
    private double min;
    private double max;

    public Uniformna(double min, double max)
    { this.min = min; this.max = max; }

    public double slBroj() { return min + (max - min) * Math.random(); }
}
```

```
// Gausova.java
public class Gausova implements Raspodela {
    private double sr;
    private double dev;

    public Gausova(double sr, double dev) { this.sr = sr; this.dev = dev; }

    public double slBroj() {
        double s = 0;
        for (int i=0; i<12; i++) s += Math.random();
        return (s - 6) * dev + sr;
    }
}
```

```
// Ekspon.java
public class Ekspon implements Raspodela {
    private double sr;

    public Ekspon(double sr) { this.sr = sr; }

    public double slBroj() { return - sr * Math.log (1 - Math.random()); }
}
```

Brojac	Srednja	Dev
-n: int = 0	-n: int = 0	-n: int = 0
+ispocetka()	-zbir: double =	-zbir: double = 0
+primi(p: Posiljka)	+ispocetka()	-zbKvad: double = 0
+dohv(): double	+primi(p: Posiljka)	+ispocetka()
	+dohv(): double	+primi(p: Posiljka)
		+dohv(): double

Min	Max
-min: double = Double::MAX_VALUE	-max: double = Double::MIN_VALUE
+ispocetka()	+ispocetka()
+primi(p: Posiljka)	+primi(p: Posiljka)
+dohv(): double	+dohv(): double

```
// Brojac.java
public class Brojac extends Pokazatelj {
    private int n = 0;

    public void ispocetka() { n = 0; }

    public void primi(Posiljka p) { n++; }

    public double dohv() { return n; }
}
```

```
// Min.java
public class Min extends Pokazatelj {
    private double min = Double.MAX_VALUE;

    public void ispocetka() { min = Double.MAX_VALUE; }

    public void primi(Posiljka p) {
        double b = raspakuj(p);
        if (b < min) min = b;
    }

    public double dohv() { return min; }
}
```

```
// Max.java
public class Max extends Pokazatelj {
    private double max = Double.MIN_VALUE;

    public void ispocetka() { max = Double.MIN_VALUE; }

    public void primi(Posiljka p) {
        double b = raspakuj(p);
        if (b > max) max = b;
    }

    public double dohv() { return max; }
}
```

```
// Srednja.java
public class Srednja extends Pokazatelj {
    private int n = 0;
    private double zbir = 0;

    public void ispocetka() { zbir = n = 0; }

    public void primi(Posiljka p) { n++; zbir += raspakuj(p); }

    public double dohv() { return zbir / n; }
}
```

```
// Dev.java
public class Dev extends Pokazatelj {
    private int n = 0;
    private double zbir = 0;
    private double zbKvad = 0;

    public void ispocetka() { zbKvad = zbir = n = 0; }

    public void primi(Posiljka p) {
        double b = raspakuj(p);
        n++; zbir += b; zbKvad += b * b;
    }

    public double dohv() {
        double sr = zbir / n;
        return Math.sqrt (zbKvad/n - sr * sr);
    }
}
```

f) Program za ispitivanje klasa statističkih pokazatelja

```
// Test.java
public class Test {
    private static Pokazatelj[] pok = {
        new Brojac(), new Min(), new Max(),
        new Srednja(), new Dev()
    };
    private static String[] vrs = { "n", "min", "max", "sr", "dev" };
    public static void main(String[] varg) {
        Analizator analiz = new Analizator();
        for (int i=0; i<pok.length; i++) analiz.prijavi(pok[i]);
        analiz.postaviRasp(new Uniformna(-10, 10));
        analiz.analiziraj(10000);
        for (int i=0; i<pok.length; i++)
            System.out.printf("%3s = %g\n", vrs[i], pok[i].dohv());
    }
}
```

```
n = 10000.0
min = -9.99998
max = 9.99790
sr = 0.0170178
dev = 5.78494
```

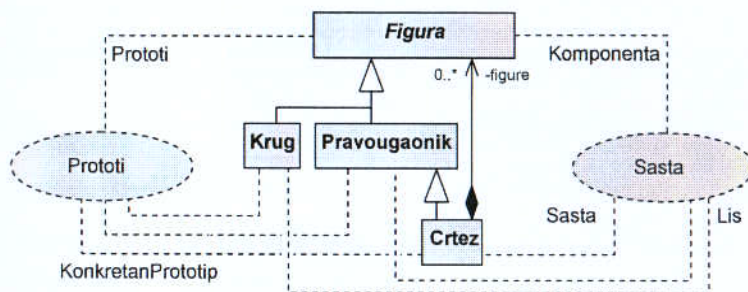

e) Projektni uzorci *Prototip i Sastav*

Figura
<pre> posId: int = 0 id: int = ++posId ==create==>+Figura(b: Boja) ==create==>+Figura() +clone(): Figura +id(): int +P(): double +pripada(T: Tacka): boolean +boja(): Boja +boja(T: Tacka): Boja raises GNePripada +dodaj(f: Figura) raises GNijeCrtez +dohvati(ind: int): Figura raises GNijeCrtez, GIndeks +brisi(ind: int) raises GNijeCrtez, GIndeks +brojFigura(): int +obilazak(): Obilazak </pre>

Krug
<pre> -r: double <<create>>+Krug(boja: Boja, r: double, C: Tacka) <<create>>+Krug(r: double, C: Tacka) <<create>>+Krug(boja: Boja) <<create>>+Krug() +r(): double +C(): Tacka +pripada(T: Tacka): boolean +P(): double </pre>

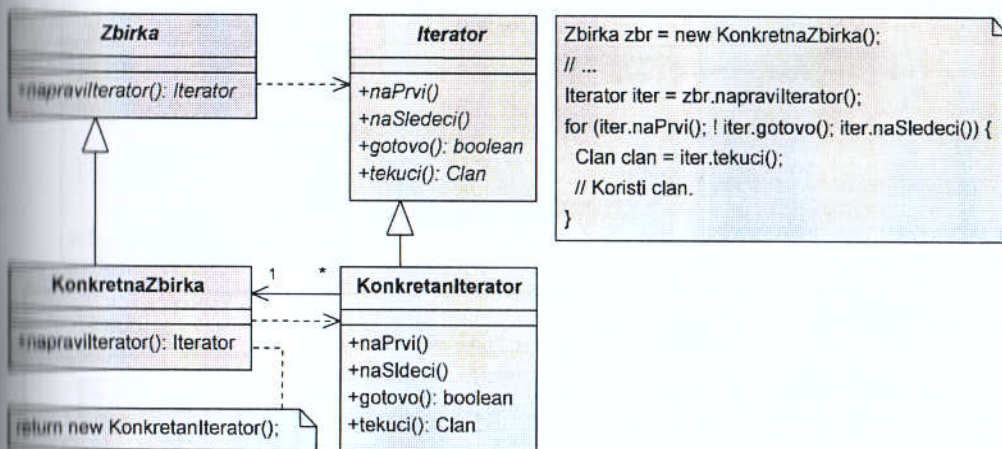
Pravougaonik
<pre> #sir: double #vis: double ==create==>+Pravougaonik(boja: Boja, A: Tacka, sir: double, vis: double) ==create==>+Pravougaonik(A: tacka, sir: double, vis: double) ==create==>+Pravougaonik(boja: Boja) ==create==>+Pravougaonik() +A(): Tacka +sir(): double +vis(): double +pripada(T: Tacka): boolean +P(): double </pre>

Crtez
<pre> -brojFigura: int <<create>>+Crtez(boja: Boja, A: Tacka, sir: double, vis: double) <<create>>+Crtez(A: Tacka, sir: double, vis: double) <<create>>+Crtez(boja: Boja) <<create>>+Crtez() +clone(): Crtez +boja(T: Tacka): Boja raises GNePripada +dodaj(fig: Figura) +dohvati(ind: int): Figura raises GIndeks +brisi(ind: int) raises GIndeks +brojFigura(): int +obilazak(): Obilazak </pre>

```
// Figura.java - Apstraktna klasa figura.
public abstract class Figura implements Cloneable {
    private static int posId = 0;
    private int id = ++posId;
    protected Boja boja;
    public Figura(Boja b) { boja = b; }
    public Figura() { boja = new Boja (); }
    public Figura clone () {
        try { return (Figura)super.clone(); }
        catch (CloneNotSupportedException g) { return null; }
    }
    public int id() { return id; }
    public abstract double P();
    public abstract boolean pripada(Tacka T);
    public Boja boja() { return boja; }
    public Boja boja(Tacka T) throws GNePripada {
        if (! pripada(T)) throw new GNePripada();
        return boja;
    }
    // Podrazumevane radnje za proste figure.
    public void dodaj(Figura f) throws GNijeCrtez
    { throw new GNijeCrtez(); }
    public Figura dohvati(int ind) throws GNijeCrtez, GIndeks
    { throw new GNijeCrtez(); }
    public void brisi(int ind) throws GNijeCrtez, GIndeks
    { throw new GNijeCrtez(); }
    public int brojFigura() { return 0; }
    public Obilazak obilazak()
    { return new PrazanObilazak(this); }
}
```

d) Projektni uzorak **Iterator** (*Iterator*)

- objektni uzorak ponašanja
- sekvencijalan obilazak članova zbirke na uniforman način
- iterator sprovodi strategiju obilaska zbirke i dohvata uzastopne članove
 - zbirka treba da podržava rad iteratora
 - iterator je često prijateljska ili čak unutrašnja klasa zbirke
 - u isto vreme može više obilazaka biti u toku nad istom zbirkom čak i po različitim kriterijumima



• **prazan iterator**

- ne radi ništa i uvek se smatra završenim
- obezbeđuje ujednačenu obradu listova i sastava u uzorku *Sastav*

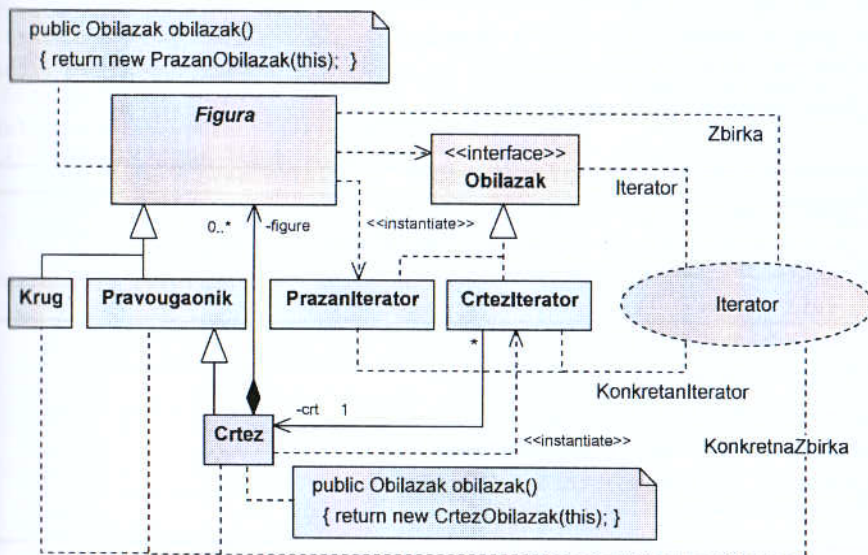
• **spoljašnji iterator**

- tok iteracija kontroliše korisnik

• **unutrašnji iterator**

- kontroliše tok iteracija
- na svakog člana zbirke primenjuje određenu radnju
- može da koristi neki spoljašnji iterator
- može da vrši selekciju među elementima
- tok iteracija može da se pre vremena prekine na zahtev korisnika

e) Primer spoljašnjeg iteratora



<<interface>> Obilazak
+naPrvu() +naPosl() +naSled() +naPret() +gotovo(): boolean +tekFigura(): Figura raises GNemaTek

PrazanObilazak
<<create>>+PrazanIterator(fig: Figura) +naPrvu() +naPosl() +naSled() +naPret() +gotovo(): boolean +tekFigura(): Figura raises GNemaTek

CrtezObilazak
-tek: int <<create>>+CrtezIterator(crt: Crtez) +naPrvu() +naPosl() +naSled() +naPret() +gotovo(): boolean +tekFigura(): Figura raises GNemaTek

```
// Obilazak.java
public interface Obilazak {
    public void naPrvu();
    public void naPosl();
    public void naSled();
    public void naPret();
    public boolean gotovo();
    public Figura tekFigura() throws GNemaTek;
}
```

```
// CrtezObilazak.java
public class CrtezObilazak implements Obilazak {
    private int tek;
    private Crtez crt;

    public CrtezObilazak(Crtez crt)
    { this.crt = crt; tek = 0; }

    public void naPrvu() { tek = 0; }
    public void naPosl() { tek = crt.brojFigura()-1; }
    public void naSled() { tek++; }
    public void naPret() { tek--; }

    public boolean gotovo()
    { return tek<0 || tek>=crt.brojFigura(); }

    public Figura tekFigura() throws GNemaTek {
        if (gotovo()) throw new GNemaTek();
        try { return crt.dohvati (tek); }
        catch (GIndeks g) { return null; }
    }
}
```

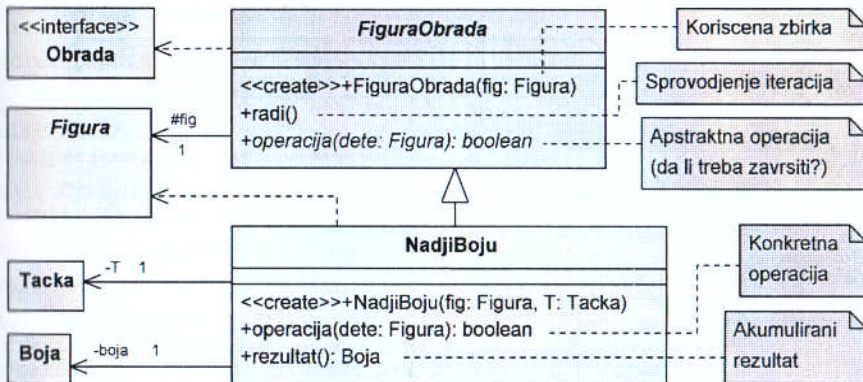
```
// PrazanObilazak.java
public class PrazanObilazak implements Obilazak {
    public PrazanObilazak(Figura fig) {}

    public void naPrvu() {}
    public void naPosl() {}
    public void naSled() {}
    public void naPret() {}

    public boolean gotovo() { return true; }

    public Figura tekFigura() throws GNemaTek
    { throw new GNemaTek(); }
}
```

f) Primer unutrašnjeg iteratora



```
// FiguraObrada.java
public abstract class FiguraObrada {
    protected Figura fig;

    public FiguraObrada(Figura fig) { this.fig = fig; }

    public void radi() {
        Obilazak obil = fig.obilazak();
        for (obil.naPosl(); !obil.gotovo(); obil.naPret())
            try { if (operacija(obil.tekFigura())) break; }
                catch (GNemaTek g) {}
    }

    protected abstract boolean operacija(Figura dete);
}
```

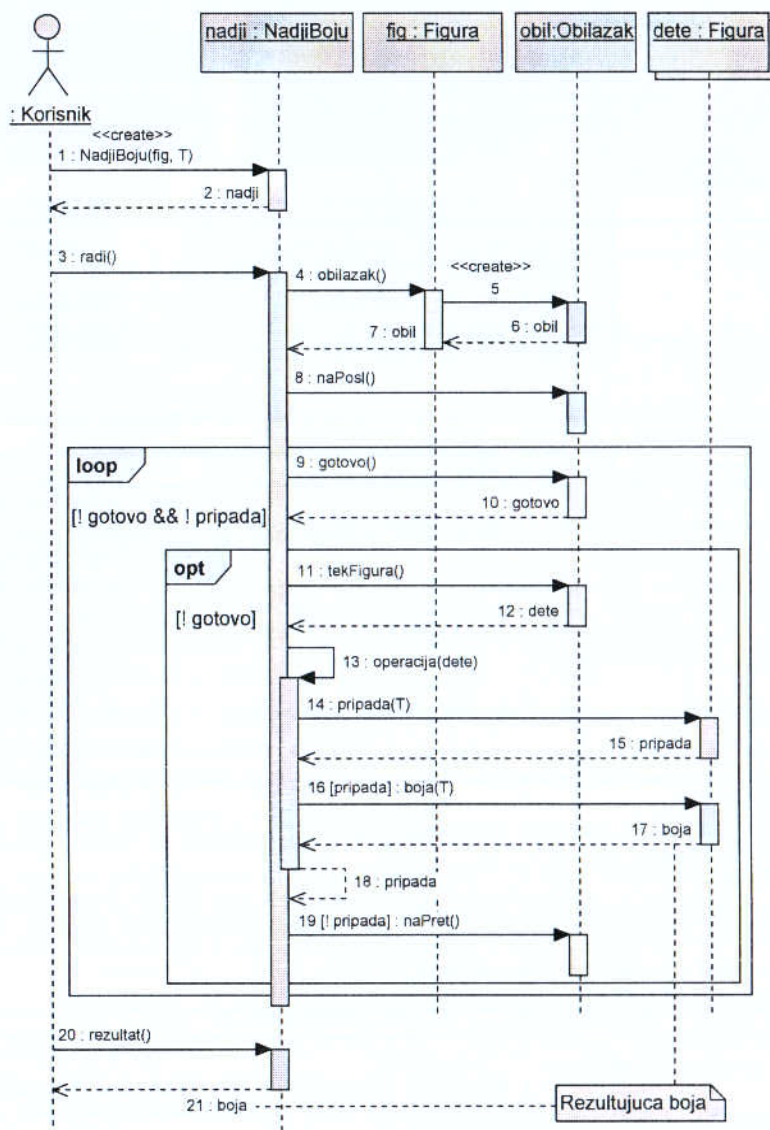
```
// NadjiBoju.java
public class NadjiBoju extends FiguraObrada {
    private Boja boja;
    private Tacka T;

    public NadjiBoju(Figura fig, Tacka T)
    { super(fig); this.T = T; }

    public boolean operacija(Figura dete) {
        if (dete.pripada(T)) {
            try { boja = dete.boja(T); } catch (GNePripada g) {}
            return true;
        } else return false;
    }

    public Boja rezultat() { return boja; }
}
```

g) Dijagram sekvence određivanja boje tačke



h) Upotreba iteratora

```
// Crtez.java
public class Crtez extends Pravougaonik {
    private int brojFigura;
    private Figura[] figure = new Figura [ 5];
    public Crtez(Boja boja, Tacka A, double sir, double vis)
    { super(boja, A, sir, vis); }

    public Crtez(Tacka A, double sir, double vis)
    { super (A, sir, vis); }

    public Crtez(Boja boja) { super (boja); }

    public Crtez() {}

    public Crtez clone () {                               // Koristi spoljašnji iterator.
        Crtez crt = (Crtez)super.clone();
        Obilazak obil = new CrtezObilazak(this);
        for (obil.naPrvu(); !obil.gotovo(); obil.naSled())
            try { crt.dodaj((Figura)obil.tekFigura().clone()); }
                catch (GNemaTek g) {}
        return crt;
    }

    public Boja boja(Tacka T) throws GNePripada {          // Koristi unutrašnji
        if (!pripada(T)) throw new GNePripada();           // iterator.
        Tacka T2 = new Tacka(T.x()-A().x(), T.y()-A().y());
        NadjiBoju nadji = new NadjiBoju(this, T2);
        nadji.radi();
        if (nadji.rezultat() != null)
            return nadji.rezultat();
        else return boja;
    }

    public void dodaj(Figura fig) {
        if (brojFigura == figure.length) {
            Figura[] nove = new Figura [ figure.length+5];
            for (int i=0; i<brojFigura; nove[i]=figure[i++]);
            figure = nove;
        }
        figure[ brojFigura++] = fig;
    }

    public Figura dohvati(int ind) throws GIndeks {
        if (ind<0 || ind>=brojFigura) throw new GIndeks();
        return figure[ ind];
    }

    public void brisi(int ind) throws GIndeks {
        if (ind<0 || ind>=brojFigura) throw new GIndeks();
        for (int i=ind; i<brojFigura-1; figure[i]=figure[i+1]);
        if (figure.length-brojFigura > 5) {
            Figura[] nove = new Figura [ figure.length-5];
            for (int i=0; i<brojFigura; nove[i]=figure[i+1]);
            figure = nove;
        }
    }

    public int brojFigura()
    { return brojFigura; }

    public Obilazak obilazak()
    { return new CrtezObilazak(this); }
}
```


i) Ostale klase

```
// Boja.java - Klasa boja.
```

```
public class Boja {
    private byte c; private byte z; private byte p;
    public Boja(byte c, byte z, byte p) { this.c = c; this.z = z; this.p = p; }
    public Boja() { c = z = p = (byte)255; }
    public byte c() { return c; }
    public byte z() { return z; }
    public byte p() { return p; }
}
```

```
// Tacka.java - Klasa tacka.
```

```
public class Tacka {
    private double x; private double y;
    public Tacka(double x, double y) { this.x = x; this.y = y; }
    public Tacka() { x = y = 0; }
    public double x() { return x; }
    public double y() { return y; }
    public double d(Tacka T) {
        return Math.sqrt(Math.pow(x-T.x,2) + Math.pow(y-T.y,2));
    }
}
```

```
// Krug.java - Klasa krugova.
```

```
public class Krug extends Figura {
    private double r; private Tacka C;
    public Krug(Boja boja, double r, Tacka C)
        { super (boja); this.r = r; this.C = C; }
    public Krug(double r, Tacka C) { this.r = r; this.C = C; }
    public Krug(Boja boja) { this (boja, 1, new Tacka()); }
    public Krug() { this (new Boja()); }
    public double r() { return r; }
    public Tacka C() { return C; }
    public boolean pripada(Tacka T) { return C.d(T) <= r; }
    public double P() { return 4 * r * r * Math.PI; }
}
```

```
// Pravougaonik.java
```

```
public class Pravougaonik extends Figura {
    protected double sir; protected double vis; private Tacka A;
    public Pravougaonik(Boja boja, Tacka A, double sir, double vis) {
        super(boja); this.A = A; this.sir = sir; this.vis = vis;
    }

    public Pravougaonik(Tacka A, double sir, double vis)
        ( this(new Boja(), A, sir, vis); )

    public Pravougaonik(Boja boja) { this (boja, new Tacka(), 1, 1);}
    public Pravougaonik() { this(new Boja()); }
    public Tacka A() { return A; }
    public double sir() { return sir; }
    public double vis() { return vis; }
    public boolean pripada(Tacka T) {
        return A.x()<=T.x() && T.x()<=A.x()+sir &&
            A.y()<=T.y() && T.y()<=A.y()+vis;
    }
    public double P() { return sir * vis; }
```

```
// GFigure.java
```

```
public class GFigure extends Exception {}
```

```
// GIndeks.java
```

```
public class GIndeks extends GFigure {}
```

```
// GNePripada.java
```

```
public class GNePripada extends GFigure {}
```

```
// GNemaTek.java
```

```
public class GNemaTek extends GFigure {}
```

```
// GNijeCrtez.java
```

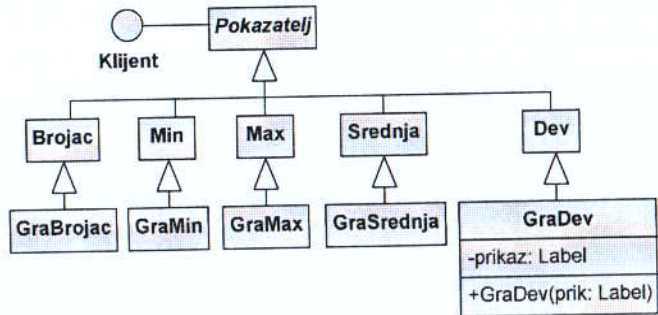
```
public class GNijeCrtez extends GFigure {}
```

Zadatak 21 Grafički statistički pokazatelji (projektni uzorak *Dopuna*)

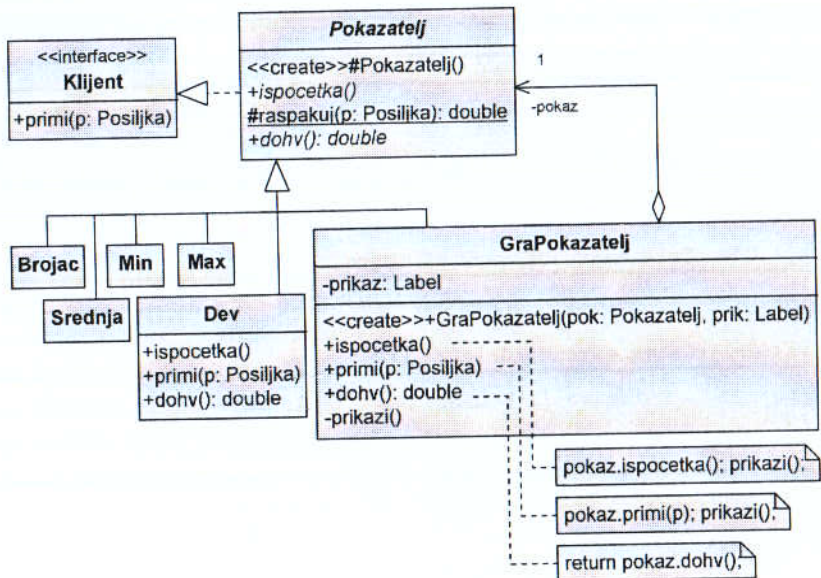
Osposobiti statističke pokazatelje iz zadatka 15 za prikazivanje svojih stanja na grafičkoj korisničkoj površi prilikom svake promene stanja.

Rešenje:

a) Loše rešenje

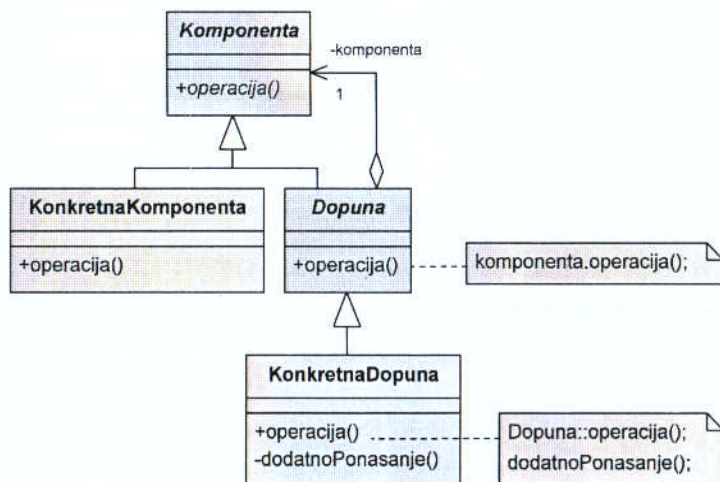


b) Dijagram klasa statističkih pokazatelja

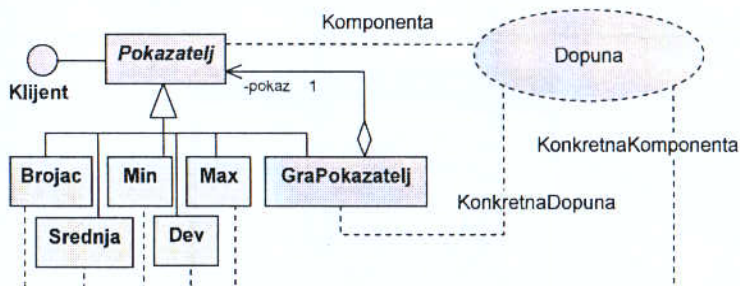


g) Projektni uzorak **Dopuna** (*Decorator*)

- objektni uzorak strukture
- dinamičko dodavanje strukturnih elemenata i ponašanja objektu
- dopuna prosleđuje zahteve dopunjenom objektu i izvršava dodatne radnje pre ili posle prosledivanja zahteva



ii) Primer uzorka *Dopuna*




```
// GraPokazatelj.java
import java.awt.Label;

public class GraPokazatelj extends Pokazatelj {
    private Pokazatelj pokaz;
    private Label prikaz;

    public GraPokazatelj(Pokazatelj pok, Label prik)
    { pokaz = pok; prikaz = prik; }

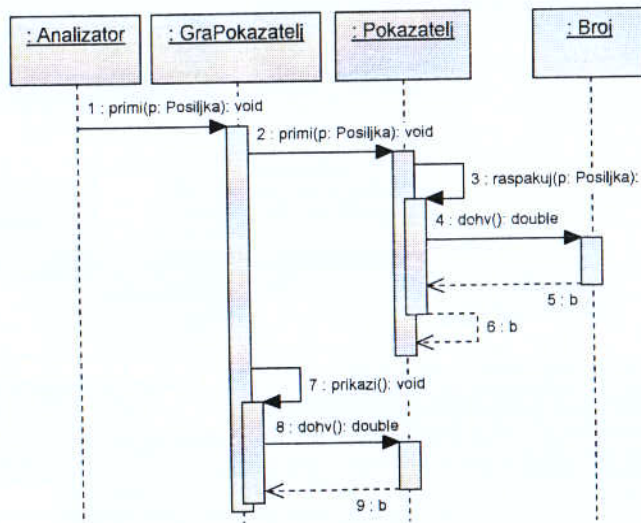
    public void ispocetka() {
        if (pokaz != null) { pokaz.ispocetka(); prikazi(); }
    }

    public void primi(Posiljka p)
    { pokaz.primi(p); prikazi(); }

    public double dohv() { return pokaz.dohv(); }

    private void prikazi ()
    { prikaz.setText(Double.toString(pokaz.dohv())); }
}
```

e) Dijagram sekvence rada grafičkog statističkog pokazatelja



f) Program za ispitivanje klasa grafičkih statističkih pokazatelja

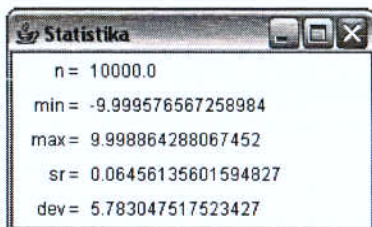
```
// Test.java
import java.awt.*;
import java.awt.event.*;

public class Test extends Frame {
    private Pokazatelj[] pok = { new Brojac(), new Min(), new Max(),
                                new Srednja(), new Dev() };
    private String[] vrs = { "n", "min", "max", "sr", "dev" };
    private Analizator analiz = new Analizator();
    { analiz.postaviRasp(new Uniformna(-10, 10)); }

    private Test () {
        super ("Statistika");
        setSize (250, 150);
        addWindowListener (new WindowAdapter() {
            public void windowClosing(WindowEvent d) { dispose(); }
        });
        popuniProzor();
        setVisible(true);
        analiz.analiziraj(10000);
    }

    private void popuniProzor() {
        Panel p1 = new Panel(new GridLayout(0,1)); add(p1, "West");
        Panel p2 = new Panel(new GridLayout(0,1)); add(p2, "Center");
        for (int i=0; i<pok.length; i++) {
            p1.add(new Label(vrs[i]+" = ", Label.RIGHT));
            Label ozn = new Label(); p2.add(ozn);
            analiz.prijavi(new GraPokazatelj(pok[i], ozn));
        }
    }

    public static void main(String[] varg) { new Test(); }
}
```



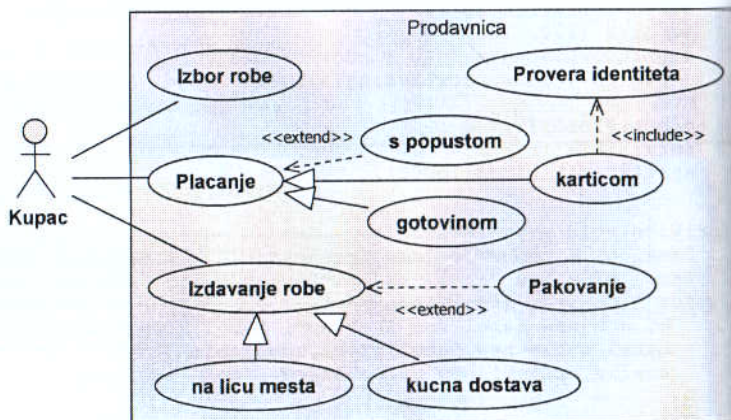
Zadatak 22 Prodavnica, fakultet i pošta (dijagram slučajeva korišćenja)

Nacrtati na jeziku *UML* dijagrame slučajeva korišćenja za:

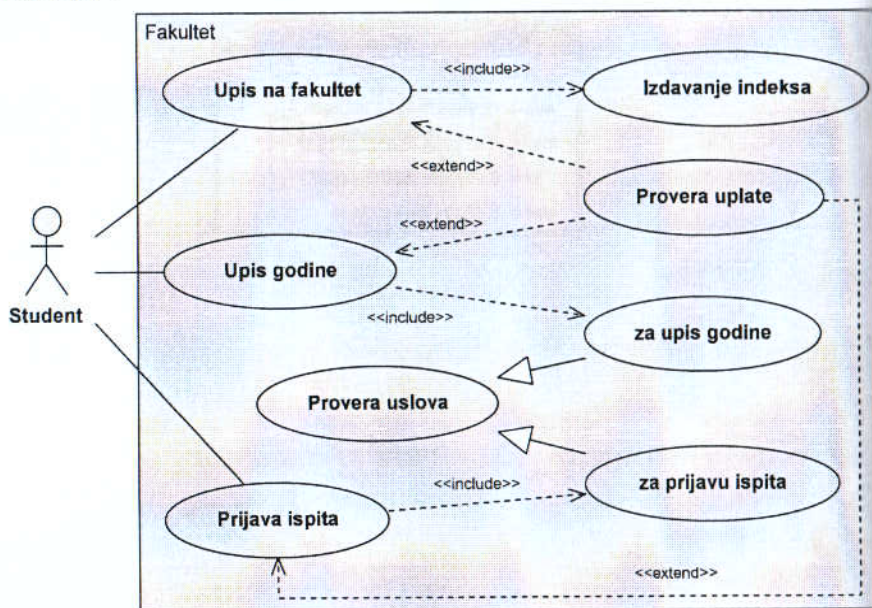
- kupovinu u prodavnici,
- rad studentske službe.
- rad poštanskog šaltera.

Rešenje:

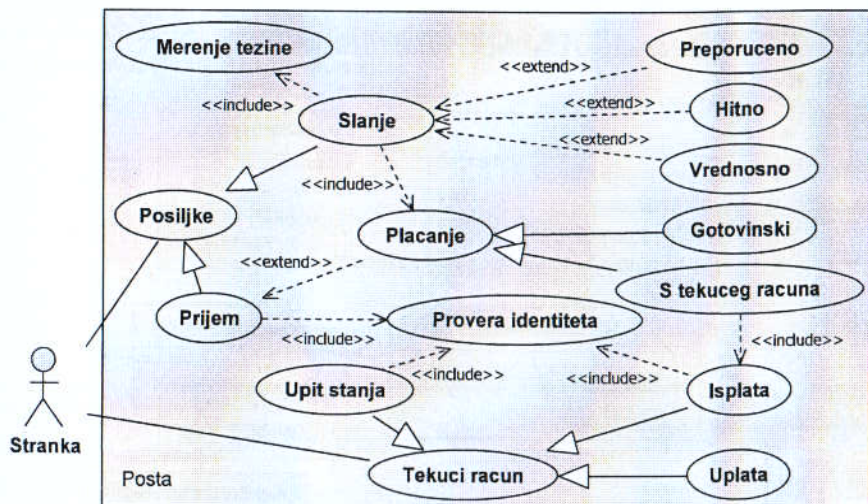
a) Kupovina u prodavnici



b) Rad studentske službe



e) Rad poštanskog šaltera

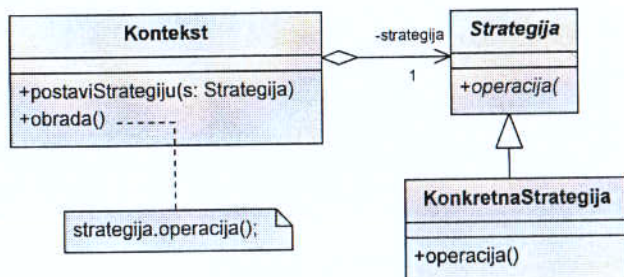
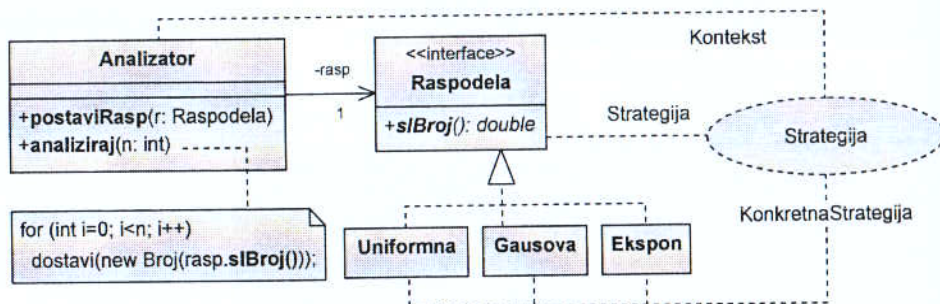


Zadatak 23 Projektni uzorci **Strategija** i **Šablonska metoda**

Uočiti projektne uzorke *Strategija* i *Šablonska metoda* u ranijim zadacima.

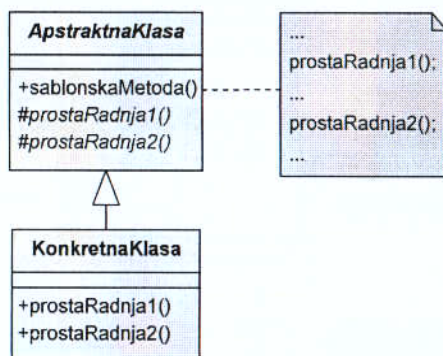
Rešenje:a) Projektni uzorak **Strategija** (*Strategy*)

- objektni uzorak ponašanja
- dinamičko podešavanje ponašanja objekta
- kontekst u toku obrade primenjuje trenutno važeću strategiju

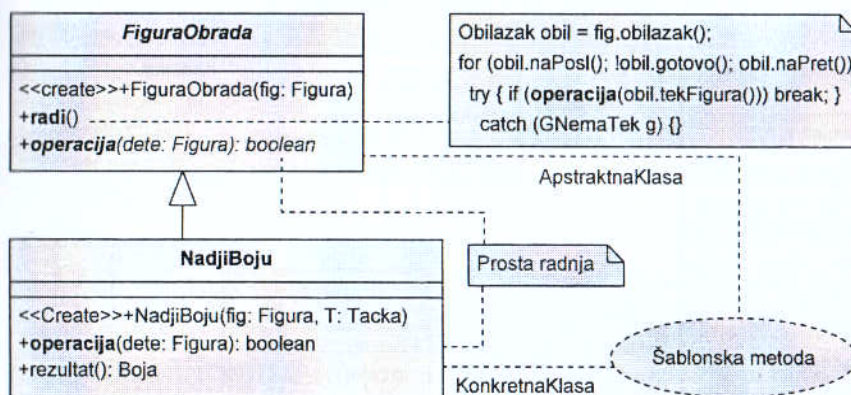
b) Primer uzorka *Strategija* u zadatku 15

g) Projektni uzorak **Šablonska metoda** (*Template Method*)

- klasni uzorak ponašanja
- statičko određivanje varijacija na određenim mestima složenog postupka
- konkretna klasa ostvaruje korake apstraktne klase

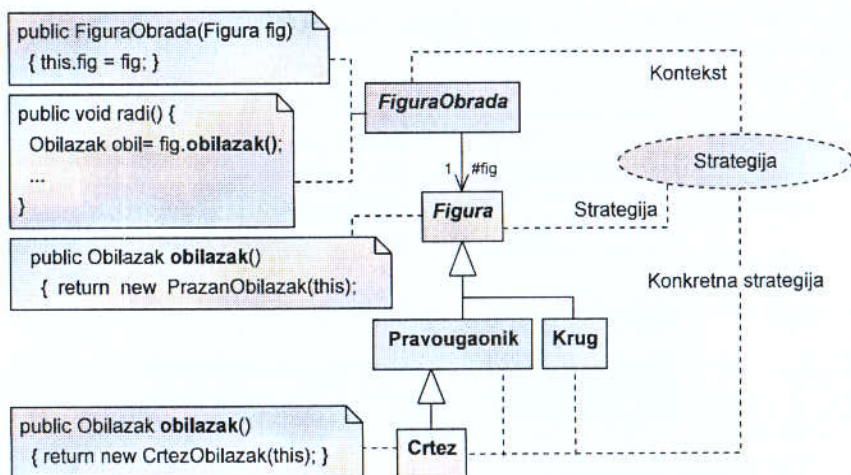
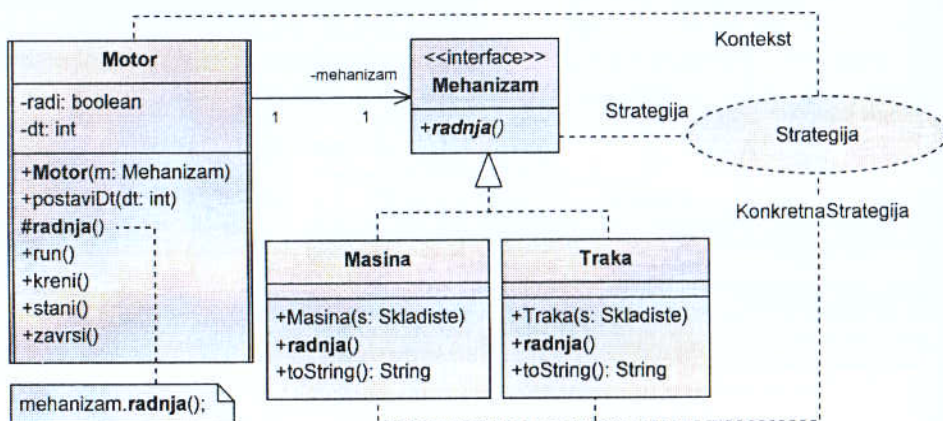


h) Primer uzorka **Šablonska metoda** u zadatku 16

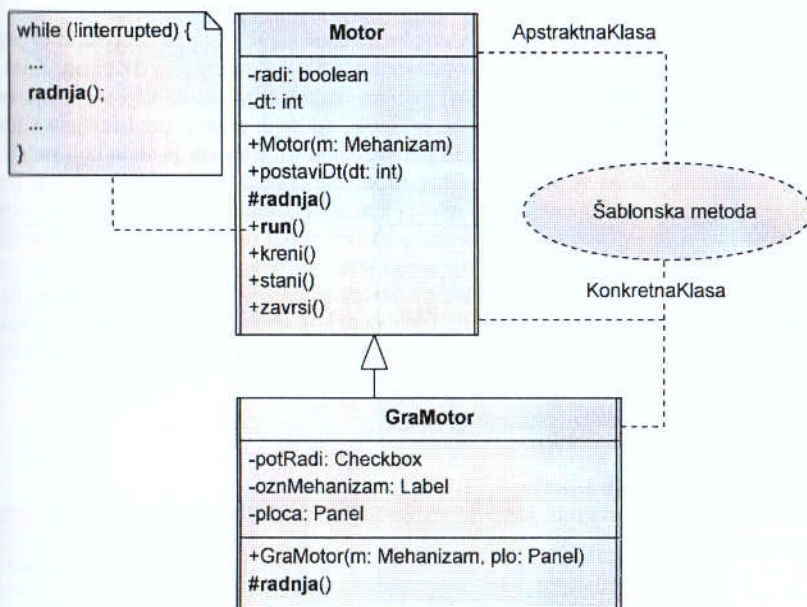


g) Zaključak

- uzorci *Strategija* i *Šablonska metoda* ostvaruju razlike u ponašanju objekata
- *Strategija* omogućava dinamičko menjanje (za vreme izvršavanja programa) ponašanja objekata
 - dati objekat u različitim trenucima može da se ponaša različito
- *Šablonska metoda* statički određuje (za vreme pisanja, odnosno prevođenja klase) razlike u ponašanju pojedinih potklasa
 - svi objekti iste klase uvek se ponašaju na isti način

f) Primer uzorka *Strategija* u zadatku 16g) Primer uzorka *Strategija* u zadatku 4

h) Primer uzorka Šablonska metoda u zadatku 4



Zadatak 24 Atomi, znak, piksel, tekst, slika i dokument {K, 01.12.2006.}

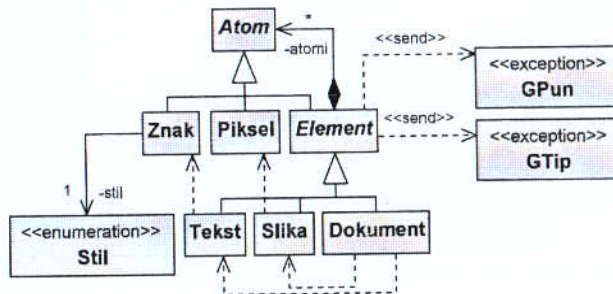
Apstraktnom atomu može da se napravi kopija (klon), da se izračuna celobrojna veličina koja predstavlja broj bajtova koje bi atom zauzeo u nekoj datoteci i atom može da se konvertuje u *String*. Znak je atom koji je opisan sa dva podatka: kodom znaka (*char*) i stilom znaka (nabrojivi tip koji uključuje vrednosti: *bold*, *italic* i *underline*). Piksel je atom opisan sa četiri bajta, od kojih prva tri predstavljaju komponente boje (crvenu, zelenu i plavu), a četvrti predstavlja faktor prozirnosti. Element je atom koji sadrži niz atoma ograničenog kapaciteta. Stvara se prazan zadatog kapaciteta, a onda mu se dodaju atomi, redom posebnom metodom. Prekoračenje kapaciteta niza atoma izaziva izuzetak. Veličina elementa određuje se izračunavanjem veličina sastavnih atoma. Tekst je element koji sadrži samo znakove i informaciju o dužini sadržaja (kratak ceo broj). Slika je element koji sadrži samo piksele, kao i informacije o širini i visini slike, izražene u broju piksela (dva kratka cela broja). Stvara se prazna zadate širine i visine, na osnovu kojih se određuje kapacitet niza piksela. Dokument je element koji ima svoje ime (*String*) i čija se veličina meri zauzećem niza znakova sa dodatnim kratkim celim brojem za opis dužine. Dokument može da sadrži samo tekstove, slike i druge dokumente.

Projektovati na jeziku UML prethodni sistem. Priložiti:

- dijagram klasa,
- prikaz korišćenih projektnih uzoraka,
- dijagram sekvence koji prikazuje scenario stvaranja jednog dokumenta sastavljenog od tekstova, slika i drugih dokumenata.

Rešenje:

a) Dijagram klasa



Atom
+kopija(): Atom +velicina(): int +toString(): String

<<enumeration>> Stil
+BOLD +ITALIC +UNDERLINE

Znak
-kod: char
<<create>>+Znak(kod: char, stil: Stil) +kopija(): Znak +velicina(): int +toString(): String

Piksel
-crvena: byte -zelena: byte -plava: byte -prozirnost: byte
<<create>>+Piksel(c: byte, z: byte, p: byte, x: byte) +kopija(): Piksel +velicina(): int +toString(): String

Element
-kap: int -brAtoma: int
<<create>>+Element(kap: int) +kopija(): Element +velicina(): int +toString(): String +dodaj(a: Atom) raises GPun, GTip #odgovara(a: Atom): boolean

<<exception>> GPun

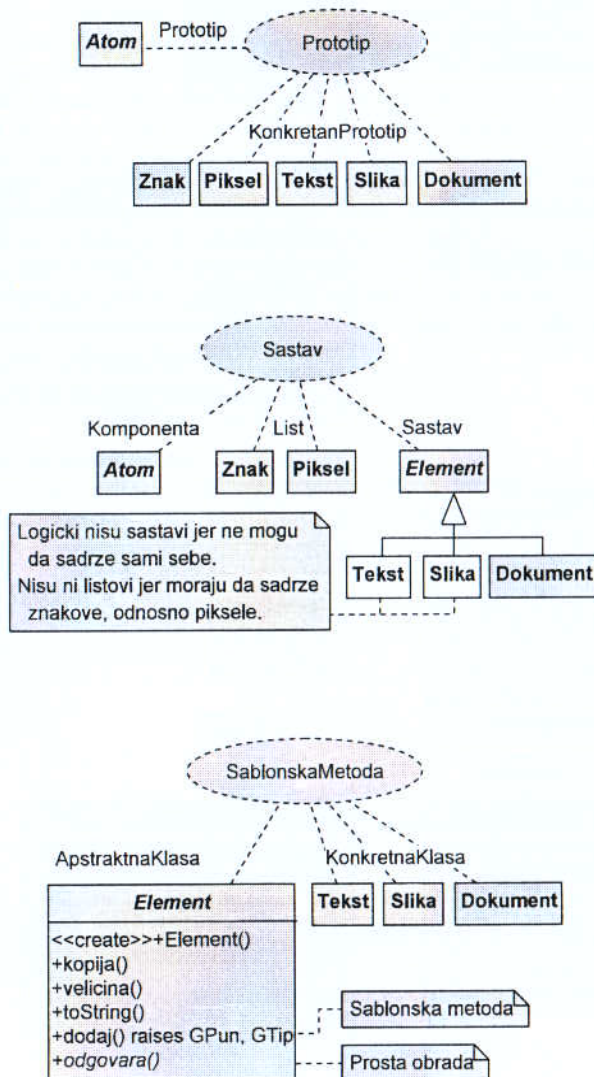
<<exception>> GTip

Tekst
-duz: short
<<create>>+Tekst(kap: int) +kopija(): Tekst +velicina(): int +toString(): String #odgovara(a: Atom): boolean

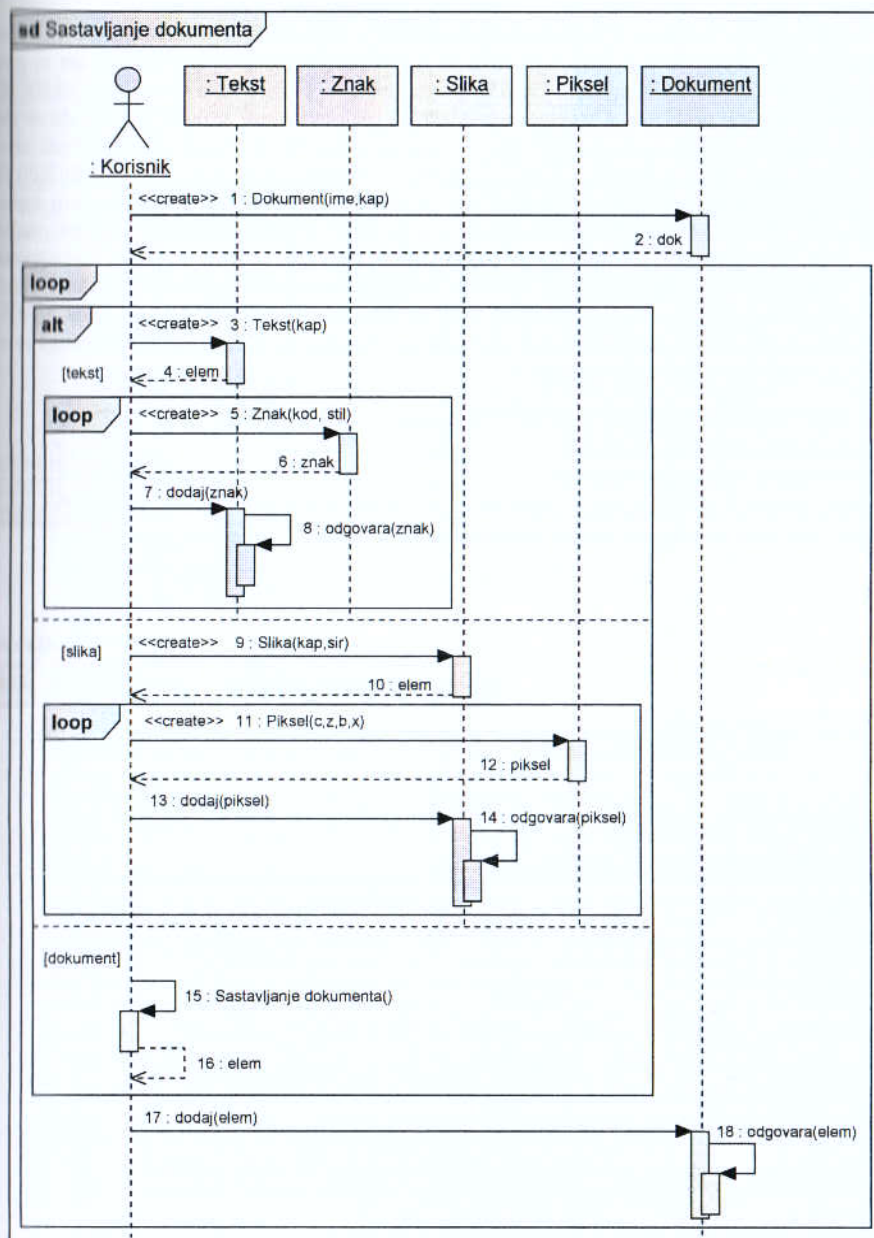
Slika
-sir: short -vis: short
<<create>>+Slika(kap: int, sir: short) +kopija(): Slika +velicina(): int +toString(): String #odgovara(a: Atom): boolean

Dokument
-ime: String -duz: int
<<create>>+Dokument(ime: String, kap: int) +kopija(): Dokument +velicina(): int +toString(): String #odgovara(a: Atom): boolean

b) Projektni uzorci



e) Dijagram sekvence sastavljanja dokumenta



Zadatak 25 Artikli, zbirke artikala, obilasci, osobe, načini plaćanja, samoposluga i kamion {K, 16.11.2008.}

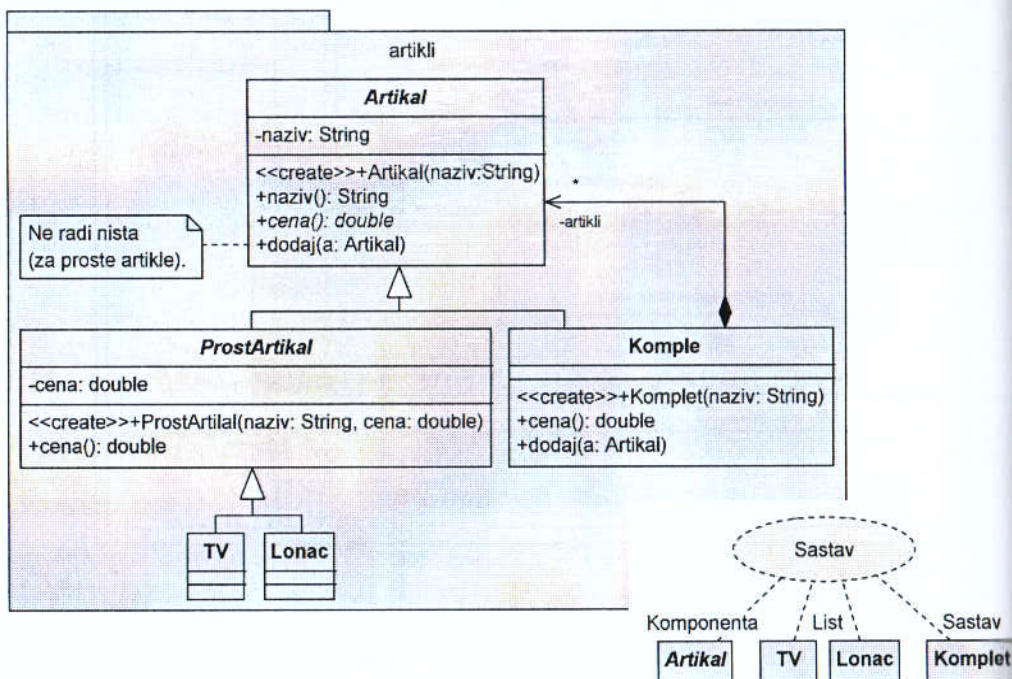
Artikal ima naziv i može da mu se odredi cena. Prost artikal je artikal koji ima zadatu cenu. Televizor i lonac su prosti artikli. Komplet je artikal koji može da sadrži proizvoljan broj artikala. Stvara se prazan posle čega se artikli dodaju jedan po jedan. Cena kompleta jednaka je zbiru cena sadržanih artikala. Zbirka artikala može da sadrži proizvoljan broj artikala. Može da se dohvati broj artikala u zbirci, da se zadati artikal doda zbirci, da se artikal sa zadatim nazivom dohvati ili izvadi iz zbirke i da se odredi ukupna vrednost svih artikala u zbirci. Obilazak zbirke radi pristupa artiklima može se obaviti po redosledu smeštanja, naziva ili cene artikala. Osoba ima ime. Kupac je osoba koja može da kupuje u zadatoj samoposluži i da plati zadatu sumu. Rezultat kupovine je zbirka kupljenih artikala. Postoje dva načina plaćanja: gotovinom ili kreditnom karticom. Prodavac je osoba koja može da se zaposli u zadatoj samoposluži i može da naplati artikle koje je zadati kupac odabrao. Samoposluga sadrži zbirku artikala koje prodaje. Može da zaposli ili otpusti zadatog prodavca, da vrati niz trenutno zaposlenih prodavaca, da odjednom nabavi više artikala, da dohvati kao i da izvadi artikal sa zadatim nazivom. Kamion samoposluge je zbirka artikala sa tekstualnim registarskim brojem.

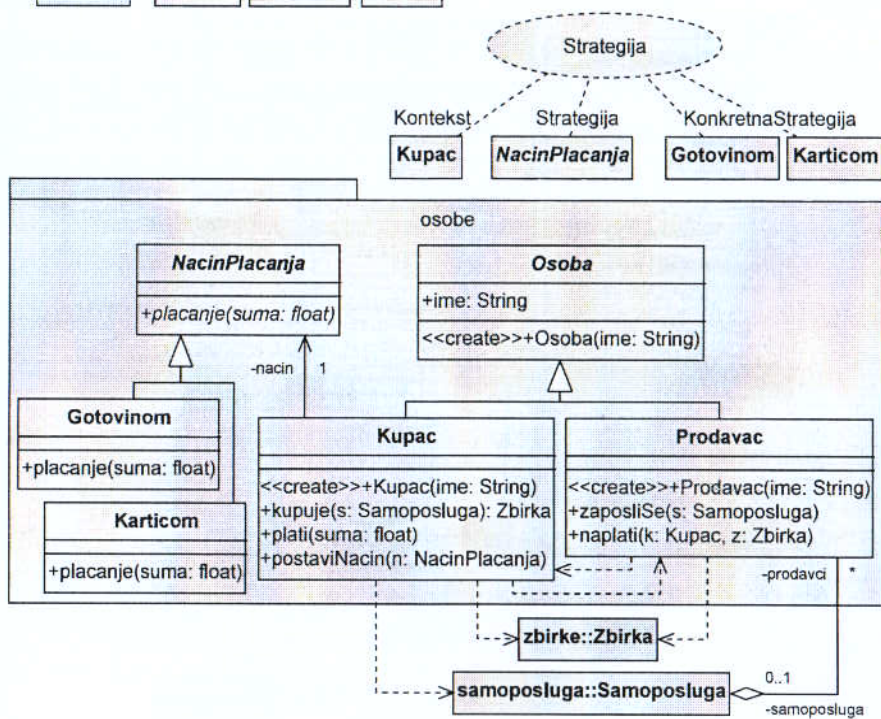
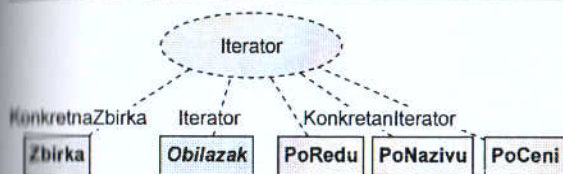
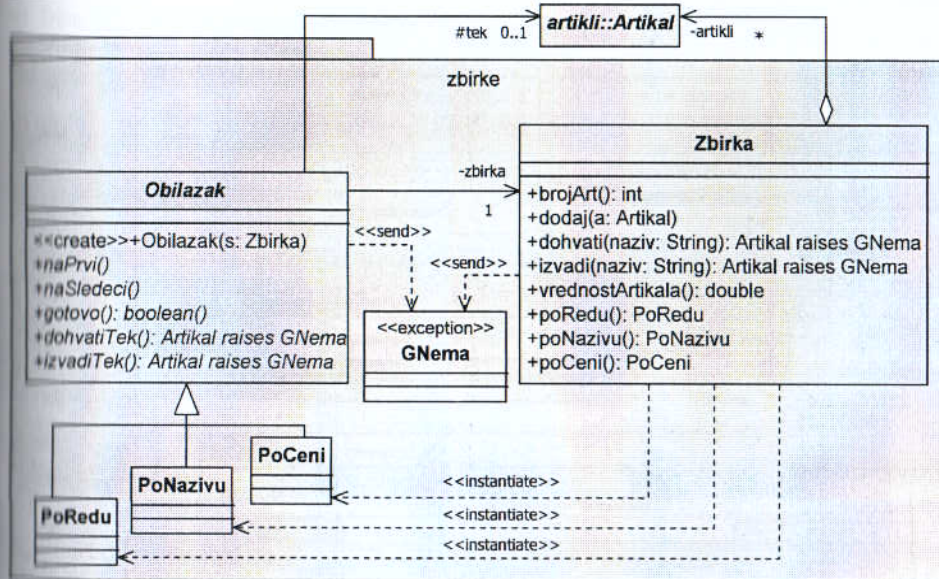
Projektovati na jeziku UML prethodni sistem. Priložiti:

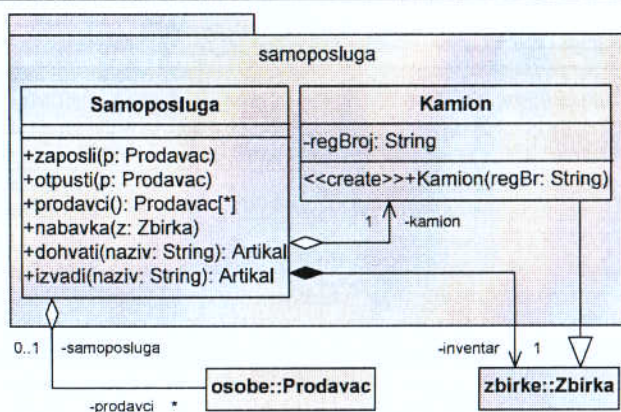
- dijagram klasa razvrstanih u pakete;
- prikaz korišćenih projektnih uzoraka;
- dijagram sekvence koji prikazuje prijem artikala u samoposluži iz natovarenog kamiona;
- dijagram komunikacije koji prikazuje prodaju artikala u samoposluži jednom kupcu.

Rešenje:

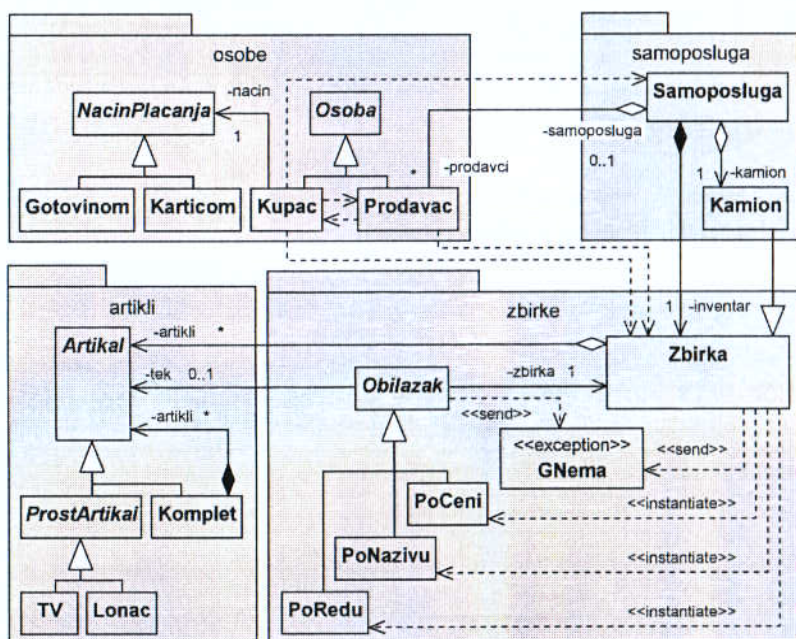
a) Dijagrami klasa i projektni uzorci



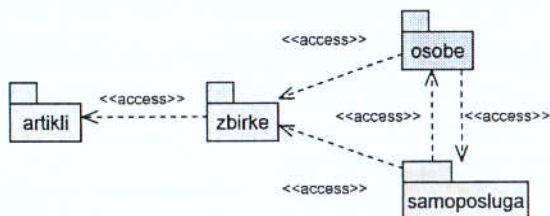




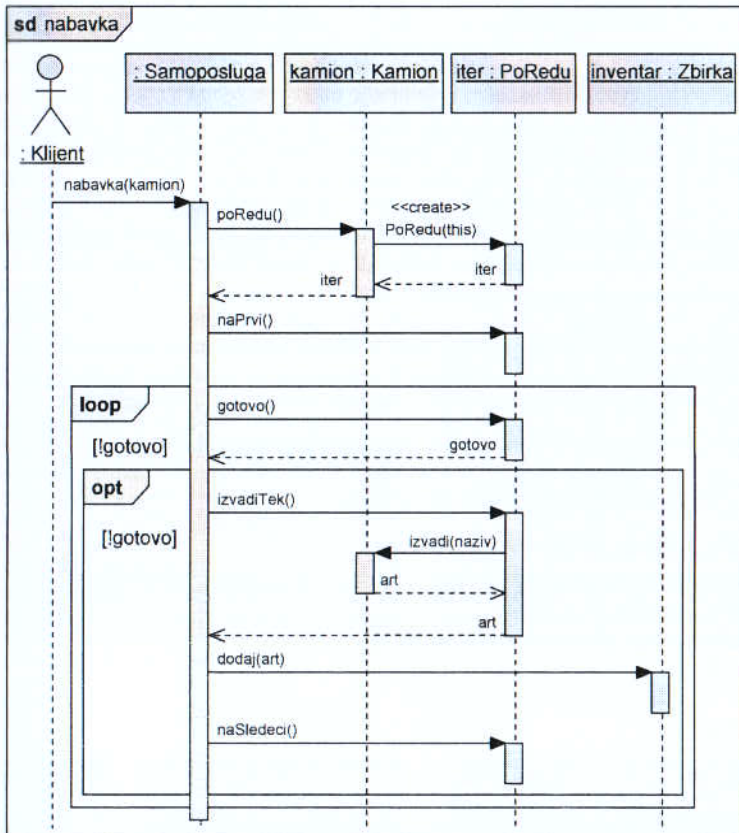
b) Sažeti dijagram klasa



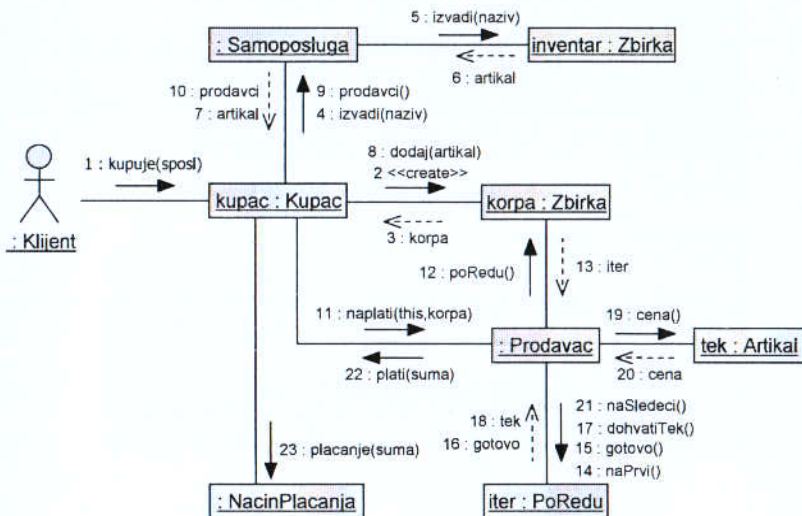
c) Dijagram paketa



d) Dijagram sekvence nabavke artikala



e) Dijagram komunikacije prodaje artikala



Zadatak 26 Polje, tabla, figure, igrači i igre {K, 30.11.2007.}

Radni okvir za igre na tabli ne zavisi od konkretne igre već obuhvata apstrakcije koje bi se koristile u raznim igrama. Figura za igre ima boju (jedno slovo) i informaciju o polju table na kojem se trenutno nalazi. Može da se dohvati jednoslovna oznaka, boja i polje na kojem se nalazi, kao i da se figura stavi na neko polje i da se premesti na drugo polje na tabli. Dužnost je figure da proveri da li je prema pravilima igre i trenutnom stanju na tabli traženo premeštanje dozvoljeno. Polja dvodimenzionalne table za igre mogu da budu prazna ili da sadrže po jednu figuru. Mogu da se dohvate koordinate polja na tabli, da se dohvati tabla kojoj pripada polje i da se dohvati figura na polju. Tabla sadrži zadati broj vrsta i kolona polja. Može da se postavi figura na dato mesto (koordinatama) na tabli, da se dohvati figura s datog mesta, da se figura premesti s jednog na drugo mesto, da se ukloni figura s datog mesta i da se dohvati polje s datog mesta. Apstraktan igrač igra figurama zadate boje na zadatoj tabli. Može da odigra potez pomerajući jednu svoju figuru na drugo mesto na tabli. Ako potez nije dozvoljen igrač treba da ponovi potez. Igra sadrži jednu tablu za igru i nekoliko igrača. Stvara se s praznom tablom zadatih dimenzija i potpunim nizom igrača. Može da se postavi početno stanje igre, da se proveri da li je igra završena i da se odigra partija. U toku partije igračima se ciklički omogućava da odigraju potez.

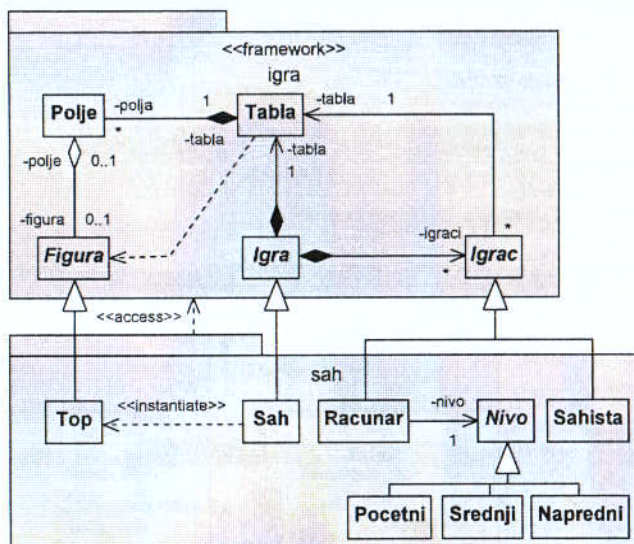
Šah je igra koju igraju dva igrača na tabli od 8x8 polja s figurama bele i crne boje. Koriste se figure za šah od kojih je jedan top. Šahista je igrač koji vuče poteze preko tastature. Računar je igrač koji automatski vuče poteze na početnom, srednjem ili naprednom nivou.

Projektovati na jeziku UML prethodni sistem. Priložiti:

- dijagram klasa razvrstanih u pakete,
- prikaz korišćenih projektnih uzoraka,
- dijagram sekvence koji prikazuje odvijanje partije.

Rešenje:

a) Dijagram klasa



<i>igra::Figura</i>
#boja: char
<<create>>+Figura(boja: char)
+oznaka(): char
+boja(): char
+stavi(na: Polje)
+pomeri(na: Polje)
+polje(): Polje
+moze(na: Polje): boolean

<i>igra::Polje</i>
-x: int
-y: int
<<create>>+Polje(t: Tabla, x: int, y: int)
+x(): int
+y(): int
+postavi(f: Figura)
+dohvatiTablu(): Tabla
+dohvatiFiguru(): Figura

<i>igra::Tabla</i>
-sir: int
-vis: int
<<create>>+Tabla(sir: int, vis: int)
+sir(): int
+vis(): int
+staviFiguru(f: Figura, x: int, y: int)
+dohvatiFiguru(x: int, y: int): Figura
+preместиFiguru(x1: int, y1: int, x2: int, y2: int)
+ukloniFiguru(x: int, y: int)
+dohvatiPolje(x: int, y: int): Polje
+isprazni()

<i>igra::Igrac</i>
#boja: char
<<create>>+Igrac(boja: char, t: Tabla)
+potez()

<i>igra::Igra</i>
<<create>>+Igra(t: Tabla, igr: Igra["*"])
+partija(): int
+pocetak()
+kraj(): boolean

<i>sah::Sah</i>
+pocetak()
+kraj(): boolean

<i>sah::Top</i>
<<create>>+Top(boja: char)
+oznaka(): char
+moze(naPolje): boolean

<i>sah::Sahista</i>
+potez()

<i>sah::Racunar</i>
+postaviNivo(n: Nivo)
+potez()

<i>sah::Nivo</i>
+odlucivanje()

<i>sah::Pocetni</i>
+odlucivanje()

<i>sah::Srednji</i>
+odlucivanje()

<i>sah::Napredni</i>
+odlucivanje()

b) Projektni uzorci

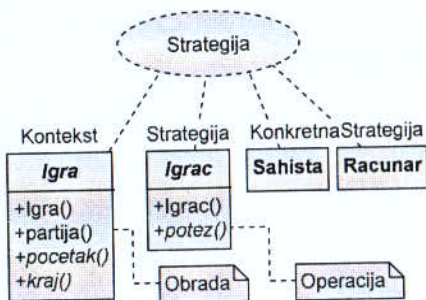
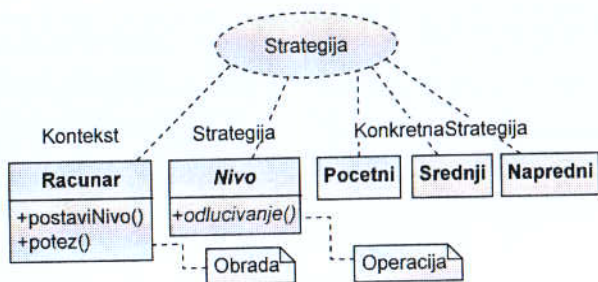
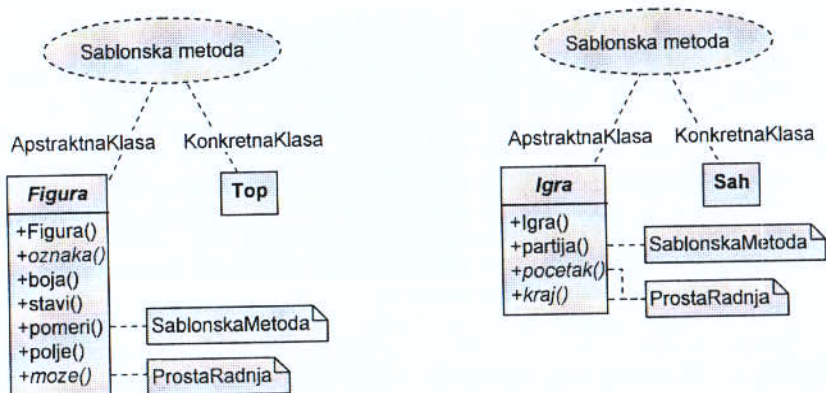
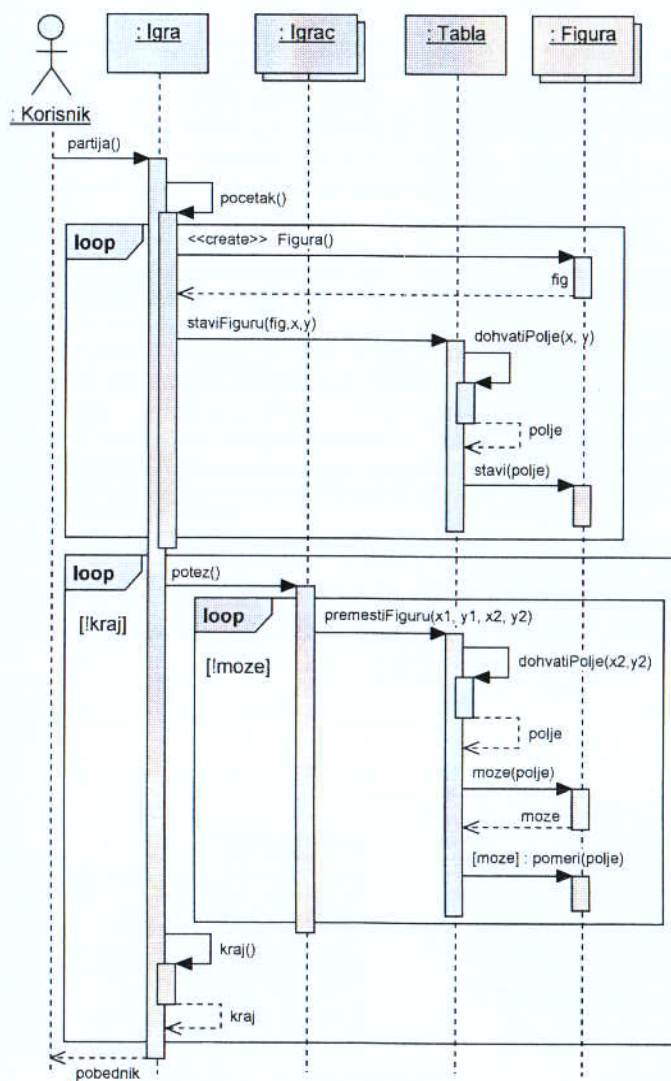


Diagram sekvence igranja partije



Zadatak 27 Uporediv, životinje, izbori, štala, obilasci štale, poslovi i osobe {K, 21.11.2009.}

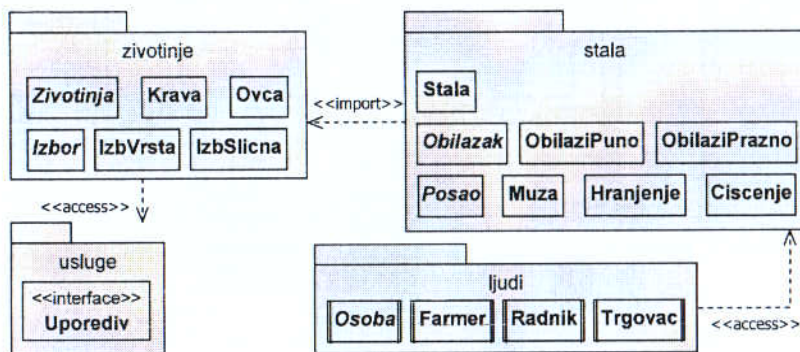
Uporediv pojam može da ispita da li je sličan drugom pojmu. Uporediva životinja ima zadatu težinu. Može da se dohvati jednoslovna oznaka vrste životinje, da se ispita sličnost s drugom životinjom (dve životinje su slične ako imaju istu oznaku vrste i iste su težine) i da se napravi klon. Krava i ovca su životinje. Oznake vrste su K, odnosno O. U štali može da boravi zadati broj (podrazumevano 10) životinja do zadate ukupne težine. Štala se stvara prazna i ne može biti više od jedne štale. Može da se uvede životinja i smesti na zadato mesto, da se pristupi životinji na nekom mestu i da se izvede životinja sa nekog mesta. Mestima za životinje u štali može da se prilazi po redosledu mesta u štali, pri čemu se nekada prilazi samo praznim mestima, a nekad samo mestima sa životinjama uz preskakanje praznih mesta. Aktivna apstraktna osoba ima ime koje može da se dohvati, ima pridruženu štalu i ciklički izvodi neku apstraktnu radnju. Ciklus rada može privremeno da se zaustavi, da se nastavi dalje i da se definitivno prekine. Farmer je osoba kojoj je pridružena i jedna životinja koja predstavlja uzorak za nabavku novih životinja. Može da nabavi novu životinju kao klon uzorka i da je smesti na prvo slobodno mesto u štali (ako životinja ne može da se smesti sačeka se dok to ne bude moglo). Radnik je osoba koja radi razne poslove oko životinja. Poslove obavlja redom po mestima u štali. Poslovi radnika su hranjenje i muža, koji se obavljaju na mestima sa životinjama, odnosno čišćenje, koje se obavlja nad praznim mestima. Trgovac je osoba koja iz određene štale može da kupuje po jednu životinju po nekom kriterijumu: prvu na koju naide zadate vrste ili onu koja je slična unapred zadatom uzorku životinje (ako takve životinje nema, sačeka da se pojavi).

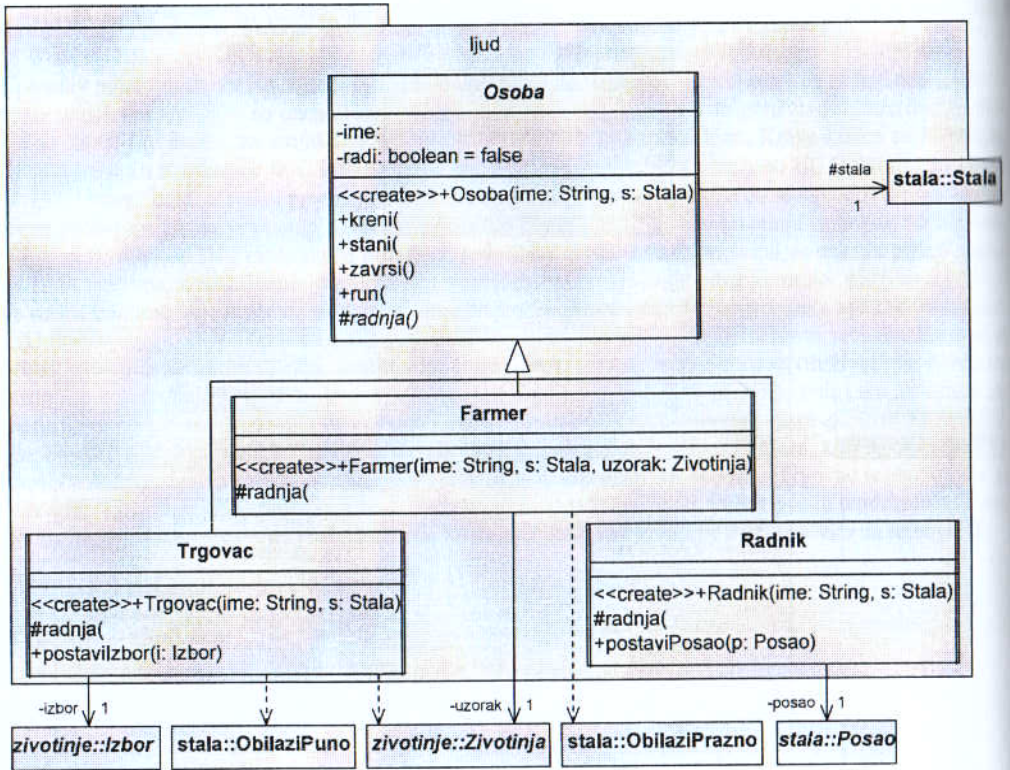
Projektovati na jeziku UML prethodni sistem. Priložiti:

- dijagram klasa razvrstanih u pakete,
- prikaz korišćenih projektnih uzoraka,
- dijagram objekata koji prikazuje farmera i štalu s jednom kravom i nekoliko ovaca.
- dijagram sekvence koji prikazuje rad farmera.

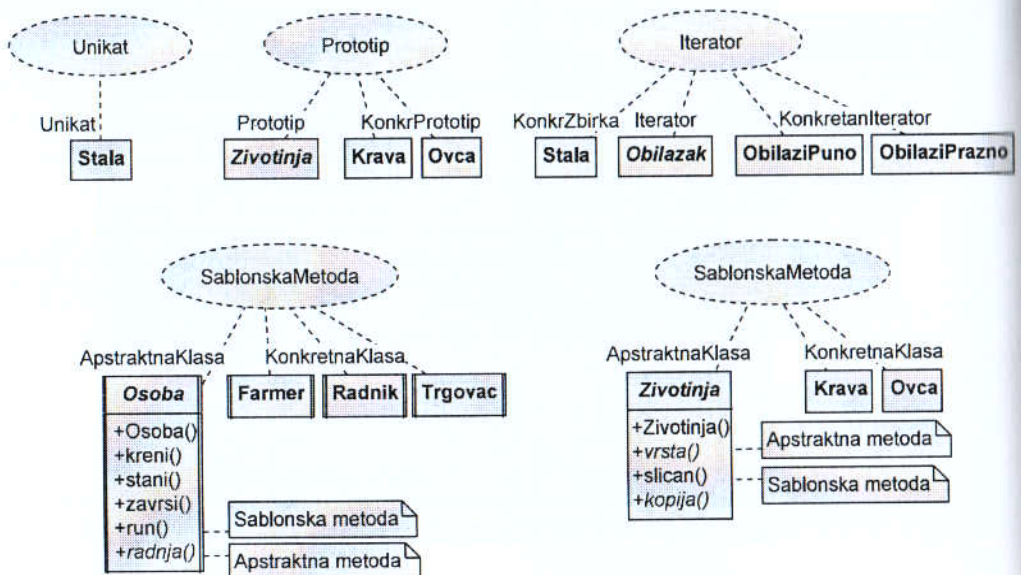
Rešenje:

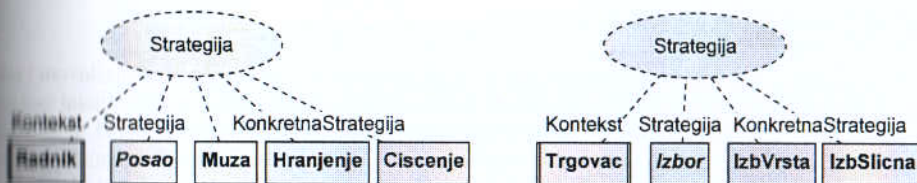
a) Dijagram paketa



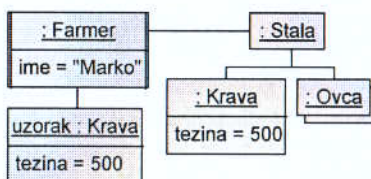


c) Projektni uzorci

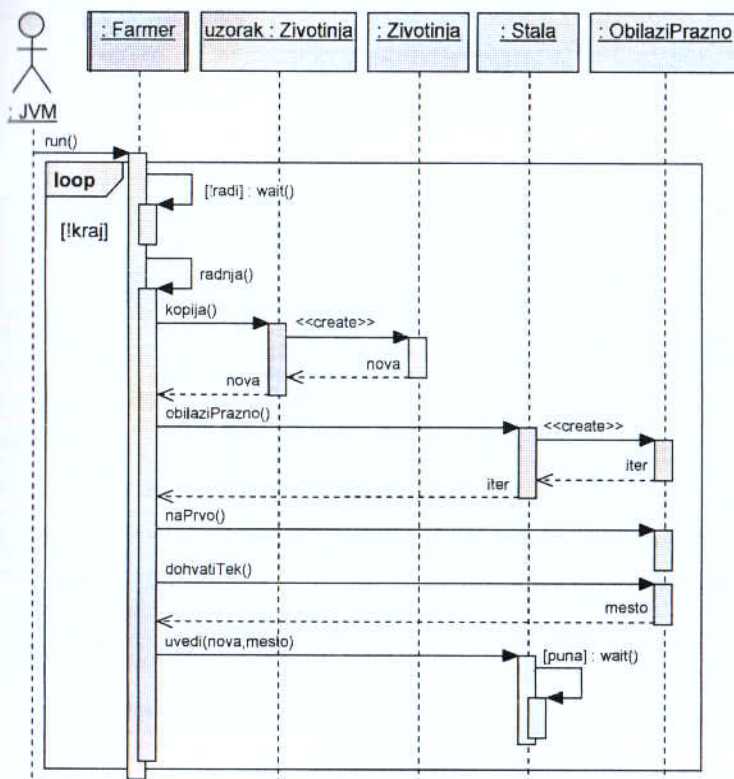




Dijagram objekata



Dijagram sekvence rada farmera



Zadatak 28 Članci, osobe, e-novine {K, 27.11.2010.}

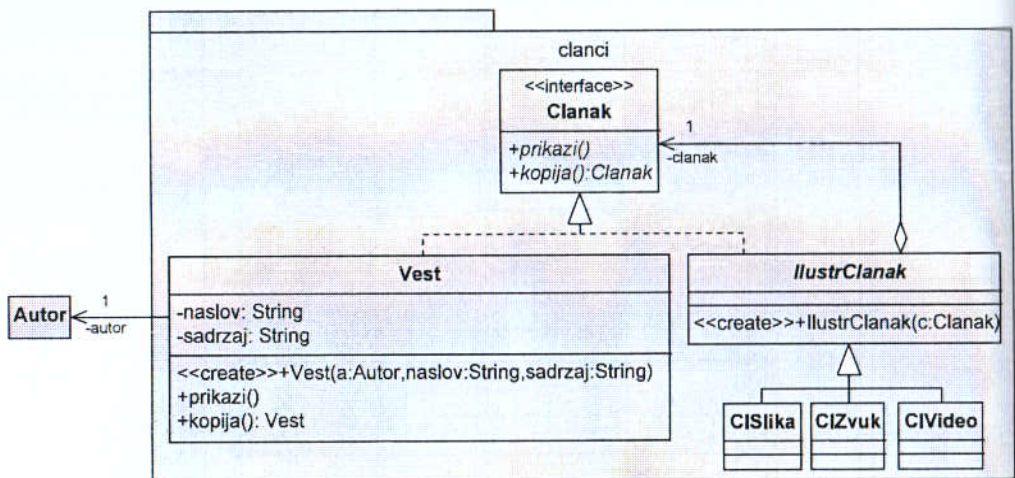
Članak može da se prikaže i da mu se napravi kopija. Vest je članak sa zadatim autorom, naslovom i tekstualnim sadržajem. Ilustrovani članak je članak koji zadatom članku dodaje jedan multimedijalni sadržaj. Članak sa slikom, članak sa zvučnim zapisom i članak s video zapisom su ilustrovani članci. Osoba ima zadato ime. Autor je osoba koja može da napiše članak za zadate elektronske novine. Pretplatnik je osoba koja može da se pretplati na zadate elektronske novine, da otkáže pretplatu, da primi dati članak koji odmah prikazuje i da od zadate elektronske novine dohvati sve članke od zadatog autora. Kolektivni pretplatnik je pretplatnik koji predstavlja više pretplatnika zadatih pri stvaranju. Elektronske novine imaju zadati naziv. Mogu da sadrže proizvoljan broj članaka koji mogu pojedinačno da se dodaju i izbacuju. Upravo dodati članak se automatski prosleđuje svim pretplatnicima. Sadržaj elektronskih novina može da se pregleda po redosledu dodavanja članaka, po redosledu autora i po redosledu naslova članaka.

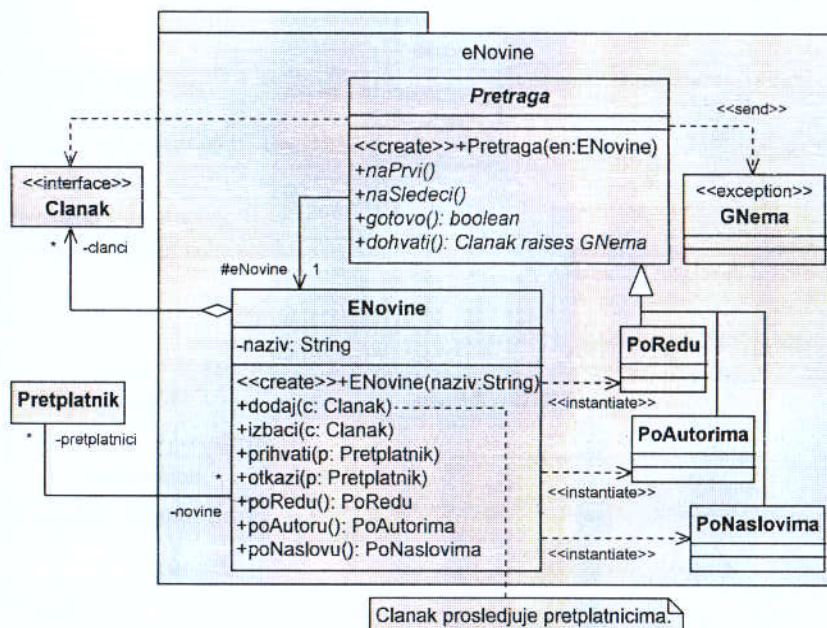
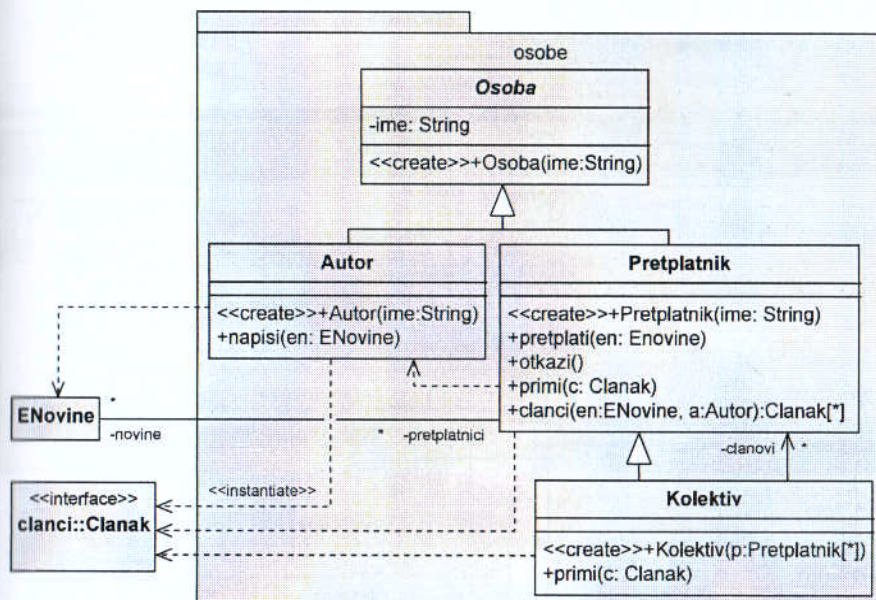
Projektovati na jeziku UML prethodni sistem. Priložiti:

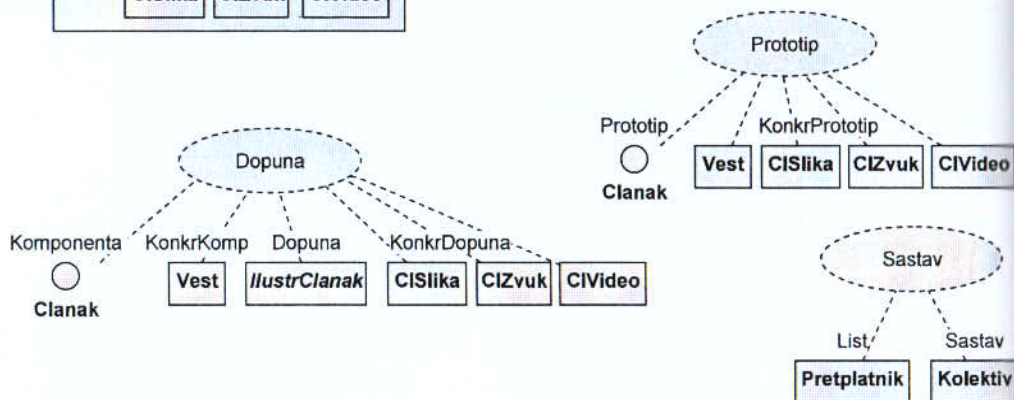
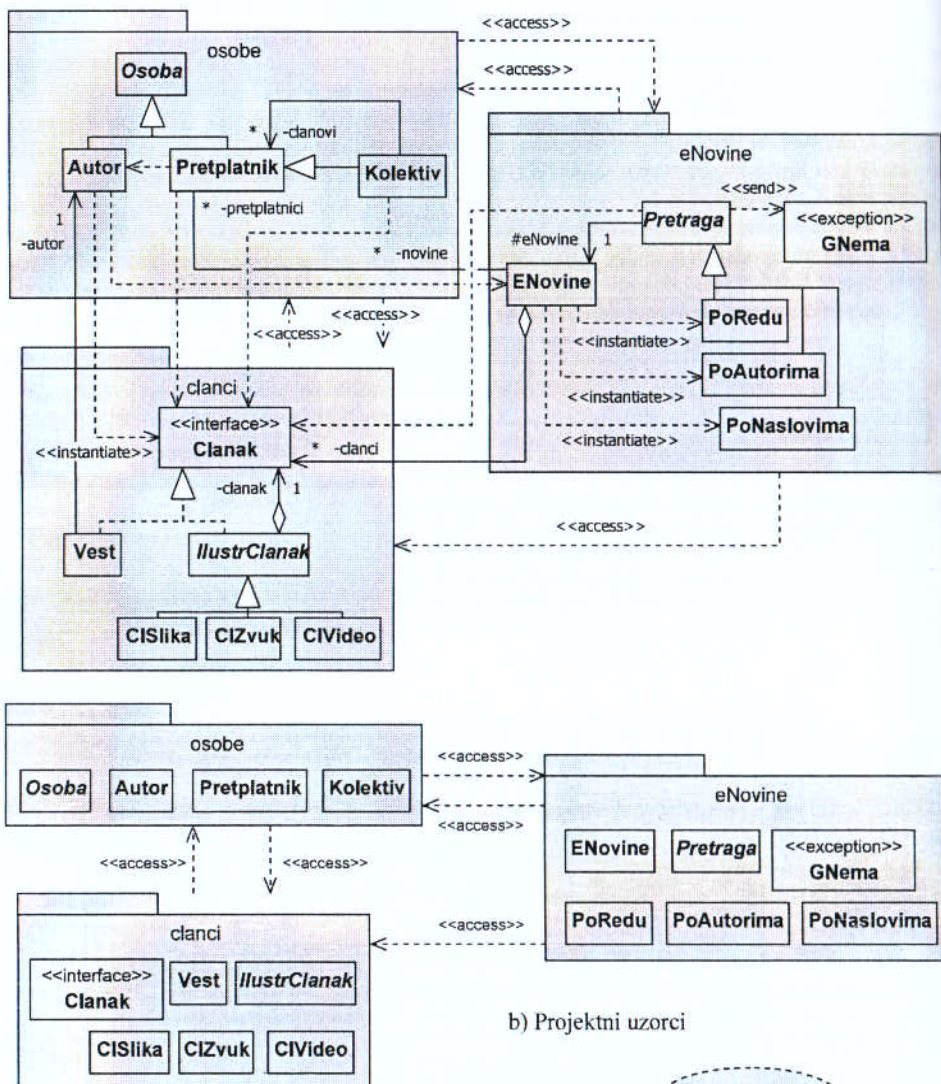
- dijagrame klasa raspoređenih u pakete (odnose među klasama i sadržaje klasa na potrebnom broju slika);
- prikaz korišćenih projektnih uzoraka;
- dijagram objekata koji prikazuje jednog pretplatnika, jednog kolektivnog pretplatnika i jedne elektronske novine sa dva ilustrovana članka od kojih jedan sadrži video zapis, a drugi upravo prispeli članak, sliku i zvučni zapis;
- dijagram sekvence koji prikazuje događaje prilikom pisanja članka sa slikom.

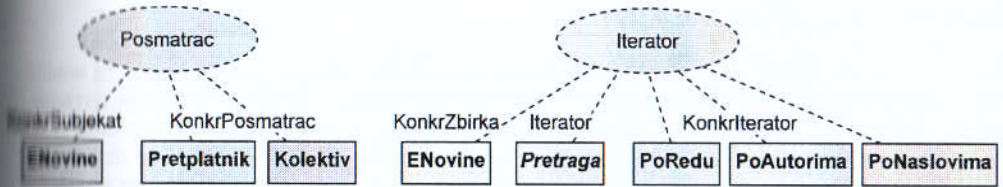
Rešenje:

a) Dijagrami klasa i paketa

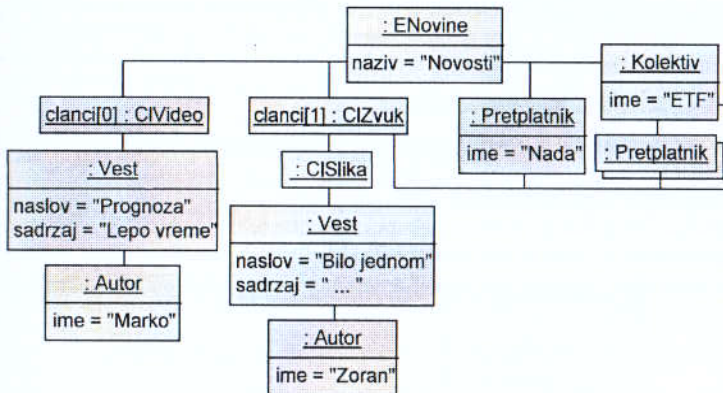




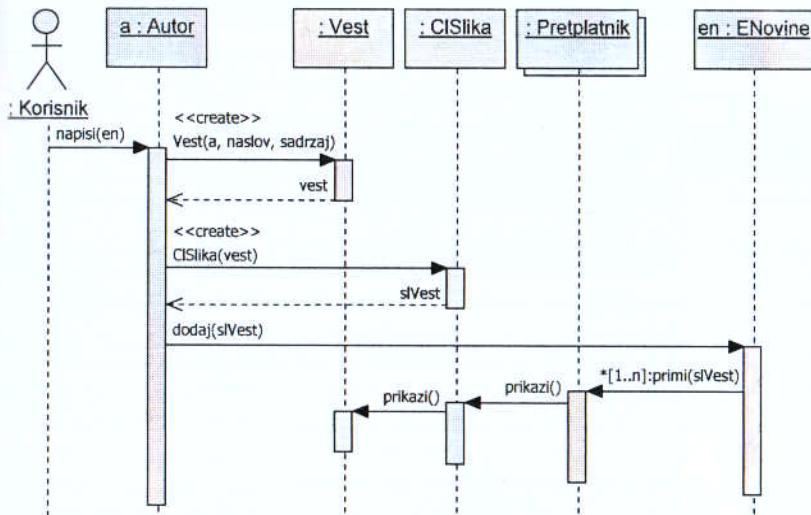




a) Dijagram objekata



b) Dijagram sekvence pisanja članka sa slikom



Zadatak 29 Berza poslova i radnici {K, 19.11.2011.}

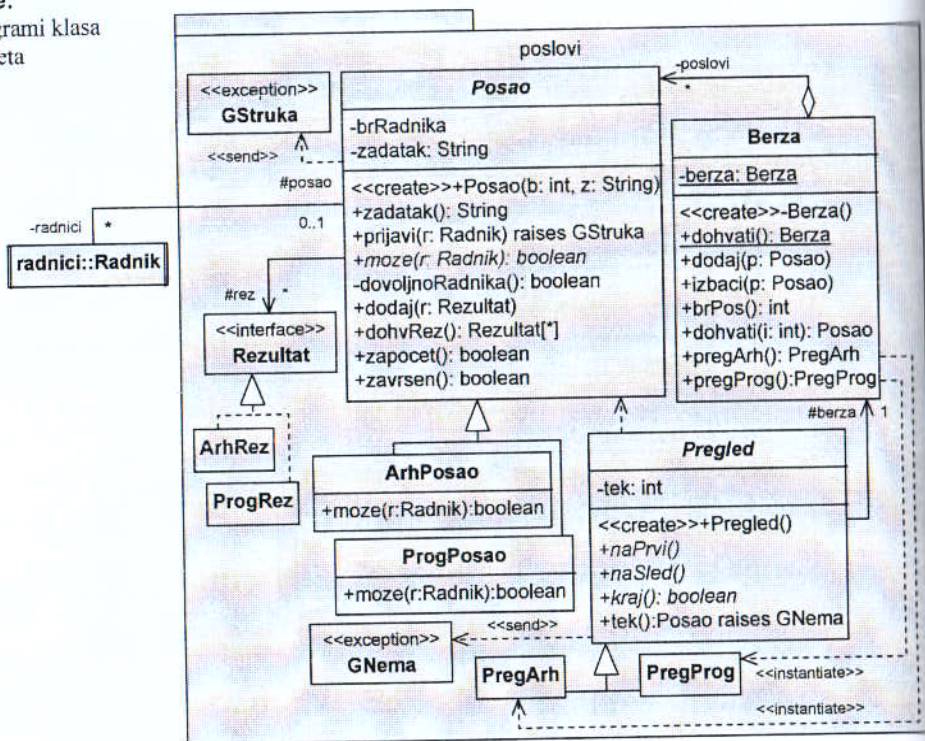
Posao je opisan potrebnim brojem radnika za njegovo izvršavanje i tekstom zadatka koji treba uraditi. Postoje programerski i arhitektonski poslovi. Posao registruje radnike koji mu se prijave za izvršenje. Greška je ako se prijavi radnik neodgovarajuće struke. Prijavljeni radnici počinju s radom kada se za posao prijavi dovoljan broj radnika. Poslu može da se doda rezultat jednog radnika, da se dohвате svi pristigli rezultati i da se ispita da li je izvršavanje posla započeto ili završeno. Jedinствена berza poslova sadrži poslove koji se dodaju i izbacuju pojedinačno. Može da se dohvati broj poslova na berzi i posao sa zadatim rednim brojem. Poslovi na berzi mogu da se redom dohвате po struci kojoj pripadaju. Aktivan radnik ima ime. Ciklički traži posao svoje struke od berze poslova, prijavljuje se pronađenom poslu, sačeka signal posla da može početi s radom, uradi svoj deo posla i dostavlja rezultat poslu. Programer i arhitekta su radnici koji izvršavaju programerske, odnosno arhitektonske poslove i stvaraju programerski, odnosno arhitektonski rezultat. Programeru se dostavlja tip posla koji može biti projektovanje, programiranje ili testiranje. Način rada programera zavisi od tipa posla.

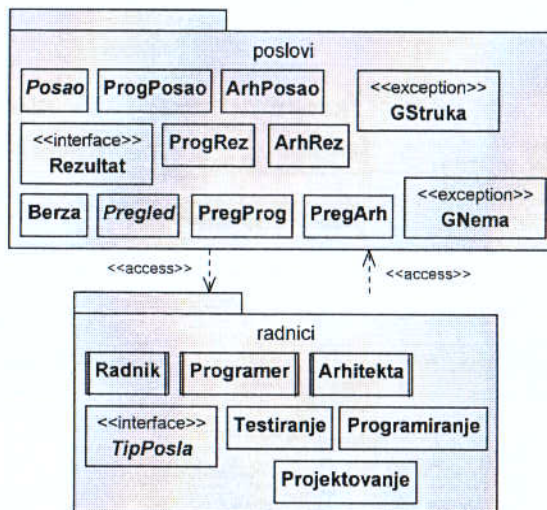
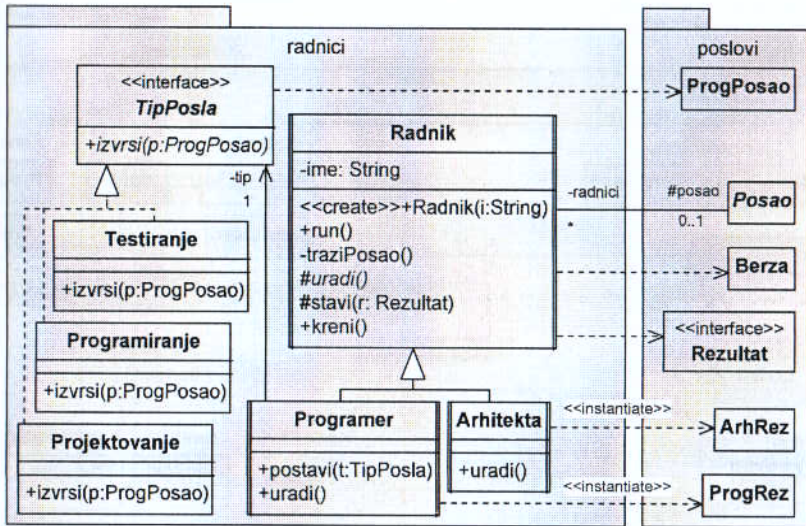
Projektovati na jeziku UML prethodni sistem. Priložiti:

- dijagrame klasa raspoređenih u pakete (odnose među klasama i sadržaje klasa na potrebnom broju slika);
- prikaz korišćenih projektnih uzoraka;
- dijagram objekata koji prikazuje berzu s jednim završenim arhitektonskim poslom sa dva radnika i jednim programerskim poslom za čije izvršavanje se prijavio jedan od potrebna tri programera;
- dijagram sekvence koji prikazuje rad jednog programera (pretpostaviti da je traženje posla na berzi uspešno).

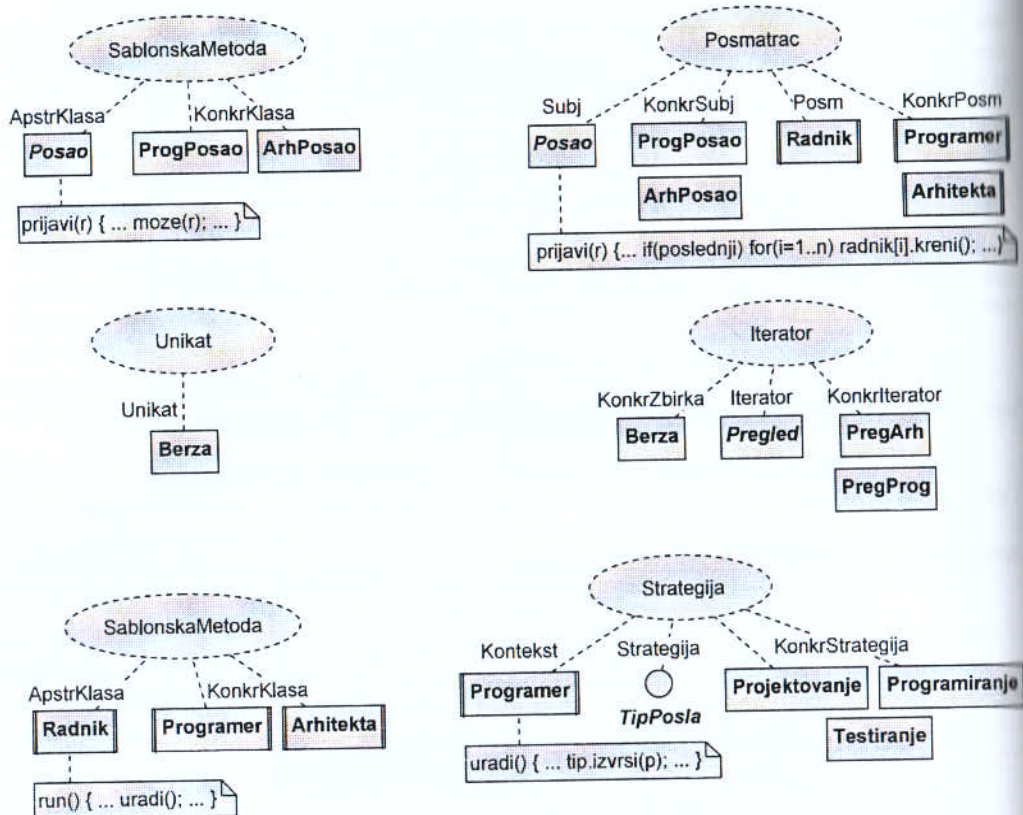
Rešenje:

a) Dijagrami klasa i paketa

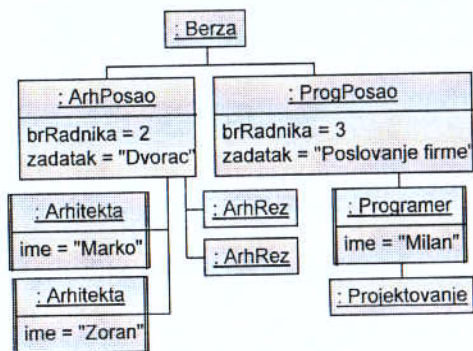




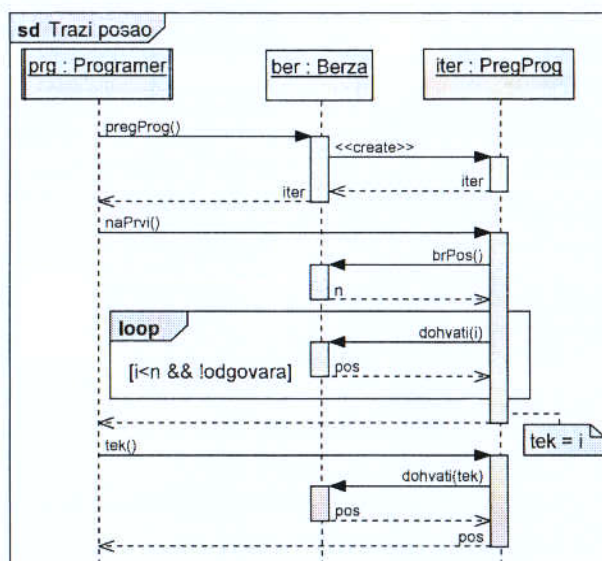
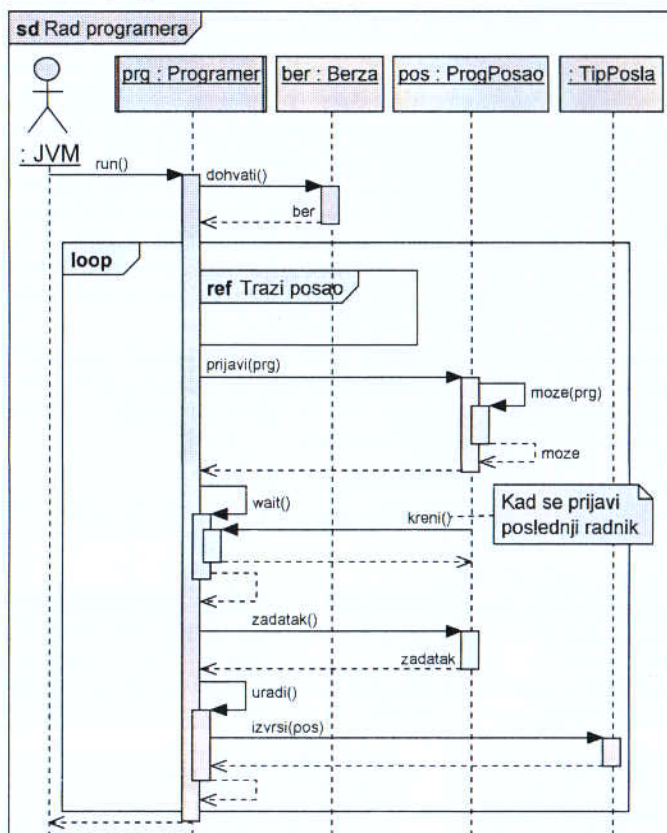
b) Projektni uzorci



c) Dijagram objekata



d) Dijagram sekvence rada programera



Zadatak 30 Geometrijske figure, geografski elementi i karte {K, 18.11.20112.}

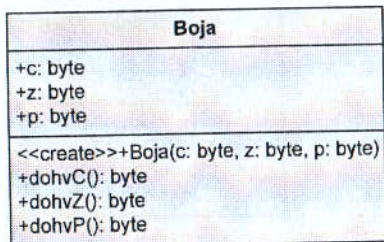
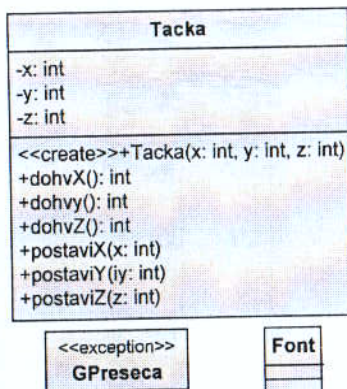
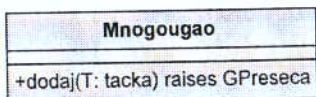
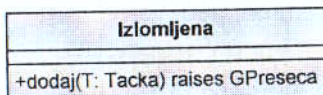
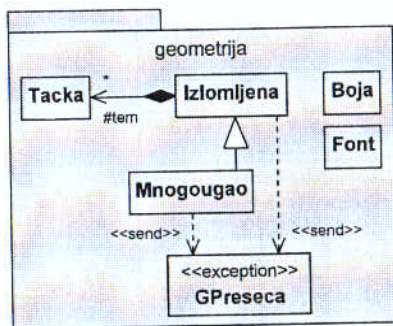
Tačka ima 3 celobrojne koordinate koje se mogu dohvatati i menjati pojedinačno. Izlomljena linija sadrži niz temena (tačaka) koja joj se dodaju redom. Mnogougao je zatvorena izlomljena linija, bez presecanja. Pri dodavanju temena greška je ako stranica između novog temena i prethodnog temena preseca neku od prethodnih stranica. Boja se predstavlja sa tri celobrojne komponente koje mogu da se dohvate. Geografski element može da se nacrtati i može mu se proslediti referenca na roditelja. Geografska karta sadrži geografske elemente koji se dodaju pojedinačno i može da se nacrtati. Izohipsa je element koji može da obuhvata druge elemente koji joj se dodaju pojedinačno. Ima visinu, mnogougao i boju unutrašnjosti mnogougla. Moguće je obični sva mesta u datoj izohipsi. Mesto je jednostavan geografski element. Osnovno mesto ima ime, poziciju (tačku), broj stanovnika i simbol mesta kojim se predstavlja na karti. To su simbol sela, varoši i grada. Mestu može da se odredi rastojanje do zadatog mesta i da se dohvati izohipsa koja ga obuhvata. Specijalno mesto je posebna vrsta mesta koja ima neke dodatne osobine u odnosu na osnovno mesto. Obeleženo mesto je specijalno mesto koje ima font kojim se ispisuje naziv mesta (detaljniji fonta zanemariti). Mesto sa okolinom je specijalno mesto koje ima mnogougao i boju kojom se mnogougao popunjava. Postoje i obeležena mesta sa okolinom.

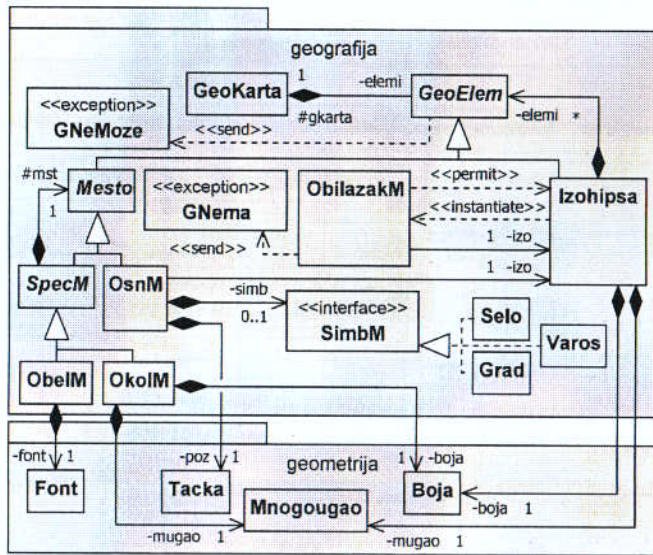
Projekтовати na jeziku UML prethodni sistem. Priložiti:

- dijagrame klasa raspoređenih u pakete (odnose među klasama i sadržaje klasa na potrebnom broju slika);
- prikaz korišćenih projektnih uzoraka;
- dijagram objekata geografske karte s izohipsom u osnovi, koja obuhvata obeleženo mesto s okolinom, koje se predstavlja simbolom grada (primere atributa navesti samo za po jedan objekat svakog tipa);
- dijagram komunikacije u kojem se za obeleženo mesto s okolinom određuje rastojanje do prvog po redu mesta unutar iste izohipse.

Rešenje:

a) Dijagrami klasa





GeoElem
+dodaj(e: GeoElem) raises GNeMoze +crtaj() +postaviRoditelja(rodit: GeoElem)

<<exception>> GNeMoze
<<exception>> GNema

GeoKarta
+dodaj(e: GeoElem) +dohvSve(): GeoElem[*] +crtaj()

Izohipsa
-vis: int <<create>>+Izohipsa(vis: int, m: Mnogougao, b: Boja) +dodaj(e: GeoElem) +obilazakMesta(): ObilazakM

ObilazakM
<<create>>+ObilazakM(i: Izohipsa) +naPrvo() +naSledce() +imaTek(): boolean +dohvTek(): Mesto raises GNema

Mesto
+rast(m: Mesto) +dohvIzo(): Izohipsa

OsnM
-ime: String -brStan: int <<create>>+OsnovnoM(ime: String, brStan: int, poz: Tacka, s: SimbM) +crtaj()

<<interface>> SimbM
+crtaj()

Selo

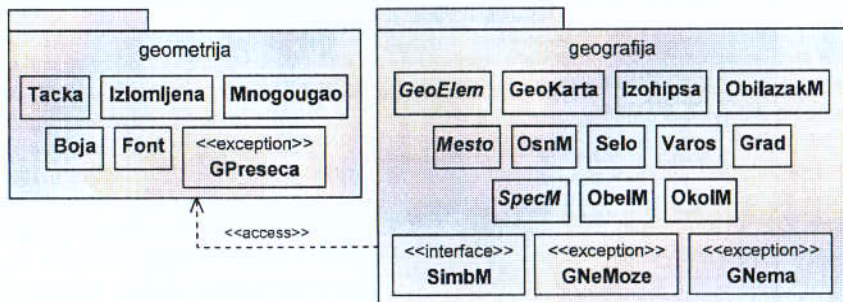
Varos

SpecM
<<create>>+SpecM(m: Mesto)

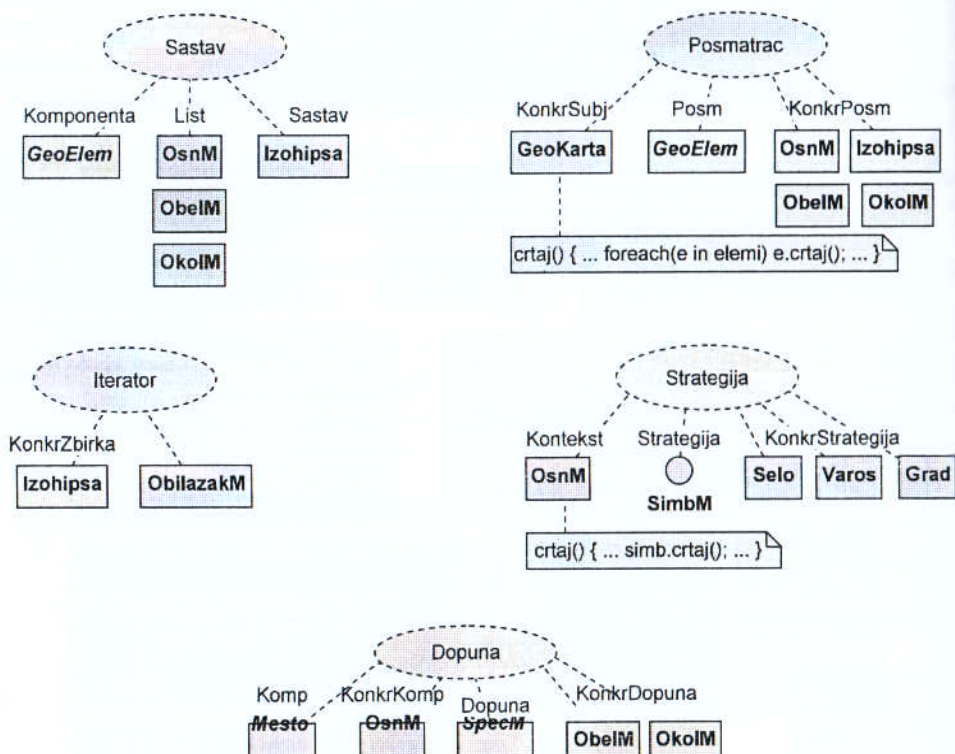
ObelM
<<create>>+ObelM(m: Mesto, f: Font) +crtaj()

OkolM
<<create>>+OkolM(m: Mesto, mu: Mnogougao, b: Boja) +crtaj()

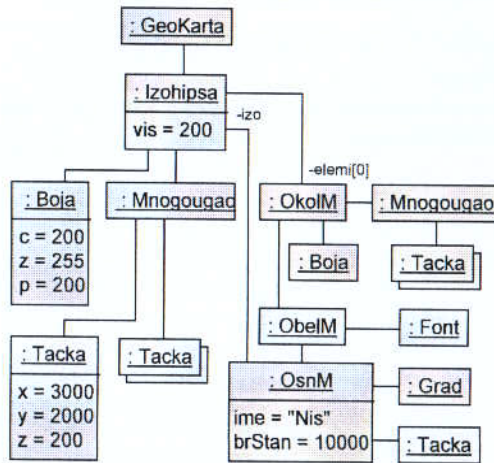
b) dijagram paketa



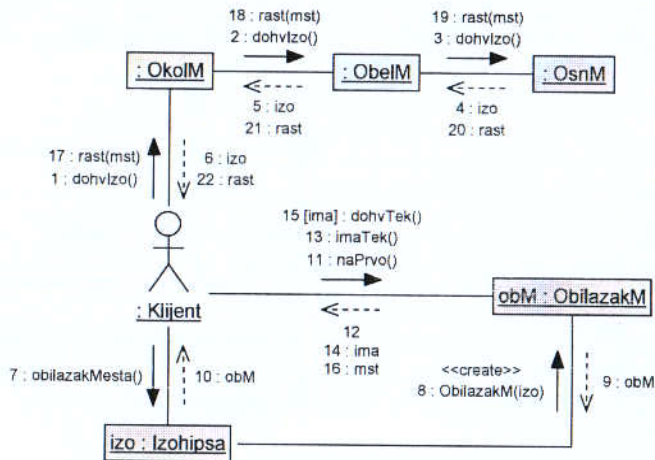
c) Projektni uzorci



d) Dijagram objekata



e) Dijagram komunikacije pri određivanju rastojanja dva mesta



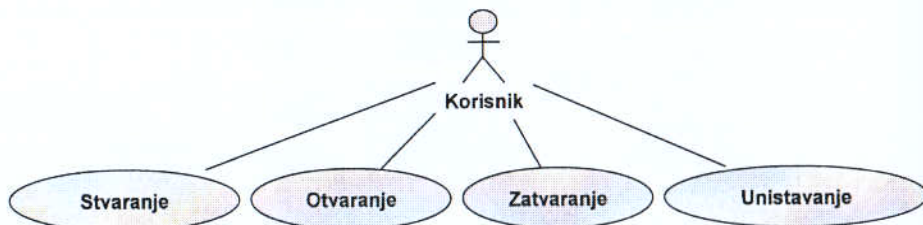
Zadatak 31 Samoposluga (dijagrami aktivnosti i stanja)

Za model samoposluge iz zadataka 5 i 14 nacrtati na jeziku **UML**:

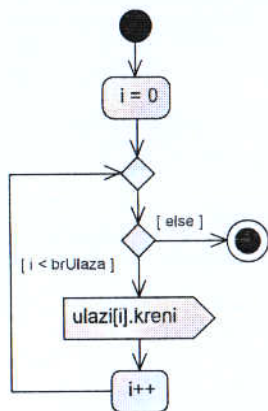
- dijagram slučajeva korišćenja,
- dijagrame aktivnosti otvaranja samoposluge, ulaza, kupca i zatvaranja samoposluge,
- dijagrame stanja kupca i samoposluge.

Rešenje:

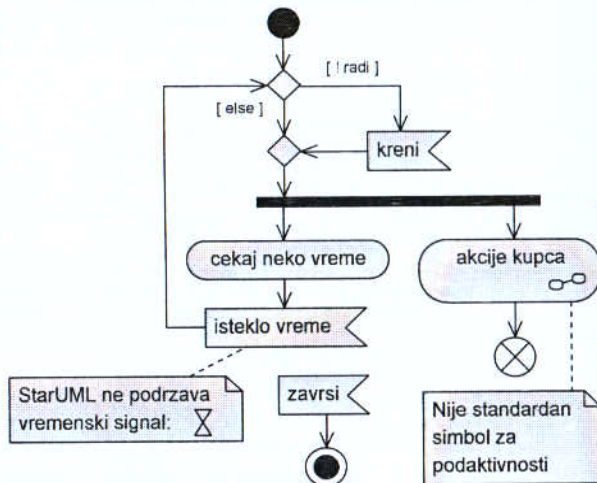
a) Dijagram slučajeva korišćenja



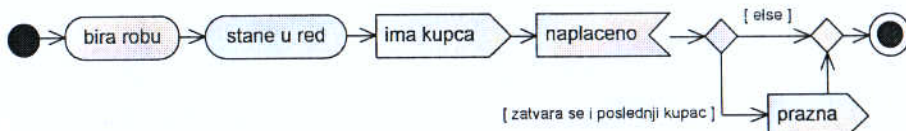
b) Dijagram aktivnosti otvaranja samoposluge



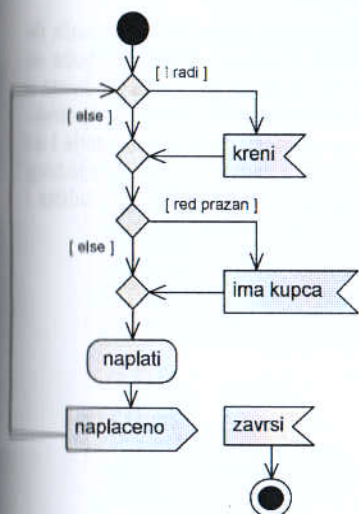
c) Dijagram aktivnosti ulaza



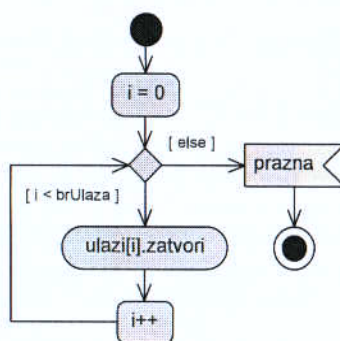
d) Dijagram aktivnosti kupca



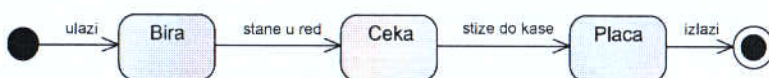
e) Dijagram aktivnosti kase



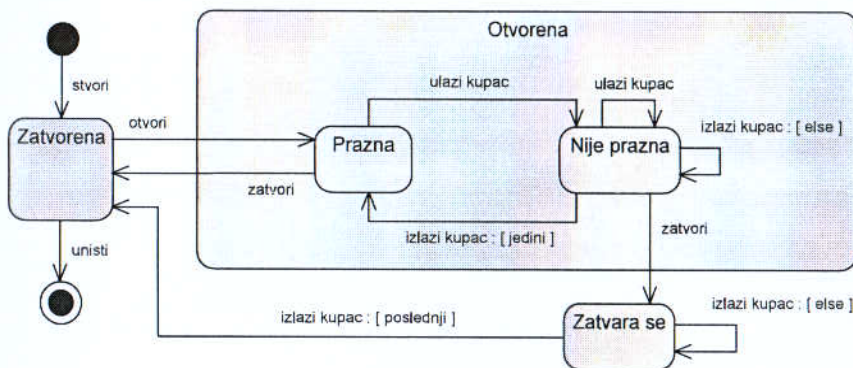
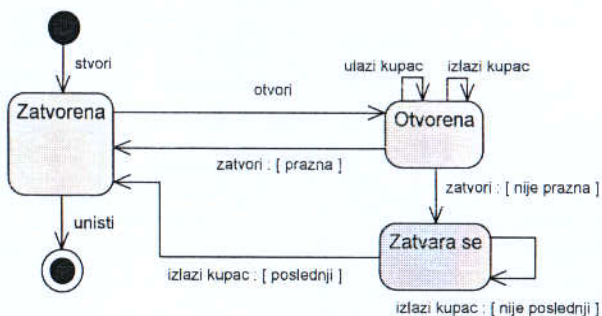
f) Dijagram aktivnosti zatvaranja samoposluge



g) Dijagram stanja kupca



h) Dijagrami stanja samoposluge



Zadatak 32 Predmeti i akteri

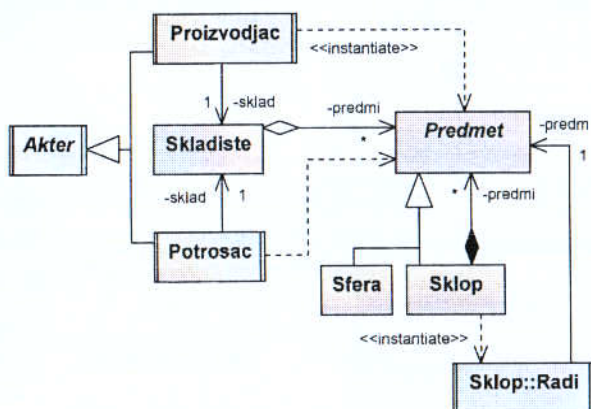
Apstraktnom predmetu može da se izračuna zapremina. Sfera je predmet. Sklop je predmet koji može da sadrži proizvoljan broj predmeta. Zapremine delova sklopa računaju se konkurentno. Skladište može da sadrži zadati broj predmeta. Predmeti se stavljaju i uzimaju jedan po jedan. Pokušaj stavljanja predmeta u puno skladište ili uzimanja iz praznog skladišta dovodi do privremenog blokiranja niti izvršioca. Apstraktni aktivan akter ponavlja neku apstraktnu akciju. Rad aktera može da se zaustavi, da se nastavi dalje i da se završi. Proizvođač je akter čija se akcija sastoji od pravljenja i stavljanja u zadato skladište po jednog predmeta. Potrošač je akter čija se akcija sastoji od uzimanja po jednog predmeta iz zadatog skladišta i izračunavanja zapremine tog predmeta.

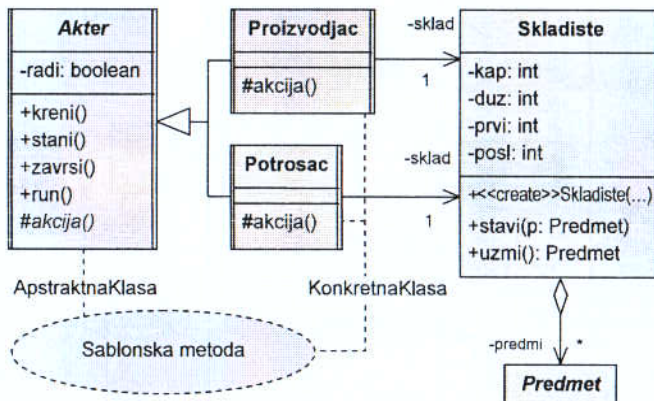
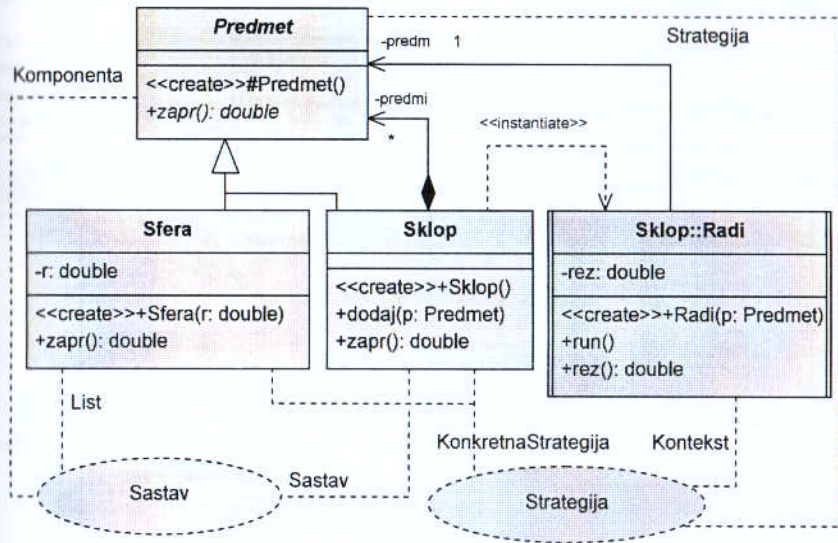
Projektovati na jeziku *UML* prethodni sistem. Priložiti:

- dijagram klasa,
- prikaz korišćenih projektnih uzoraka,
- dijagrame interakcije rada proizvođača i potrošača,
- dijagrame aktivnosti proizvođača i potrošača,
- dijagram aktivnosti računanja zapremine sklopa,
- dijagrame stanja skladišta, proizvođača i potrošača.

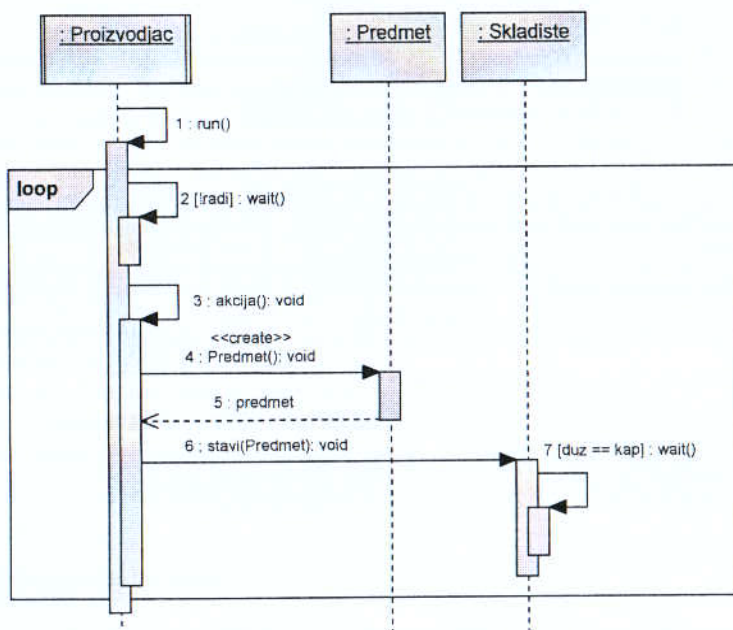
Rešenje:

a) Dijagrami klasa i projektni uzorci

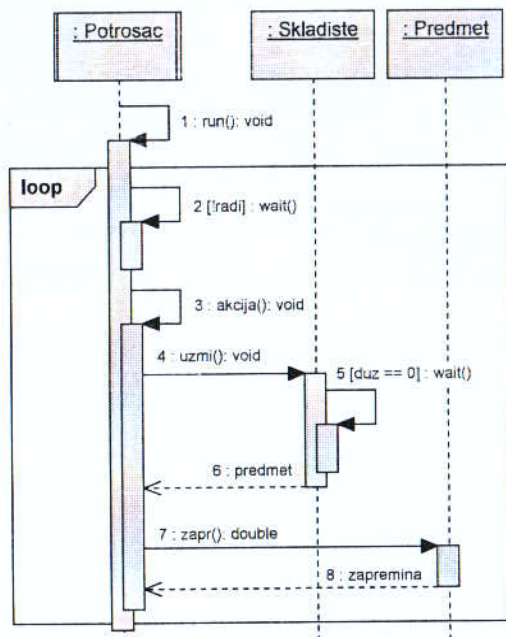




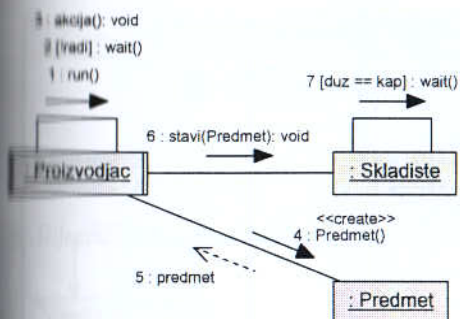
b) Dijagram sekvence proizvođača



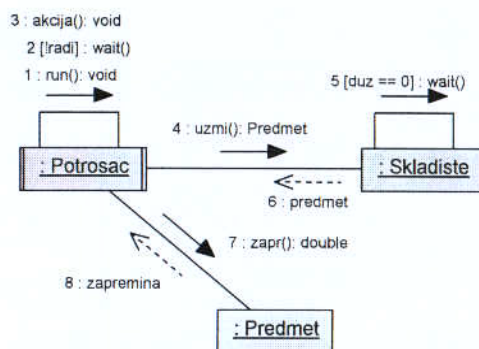
c) Dijagram sekvence potrošača



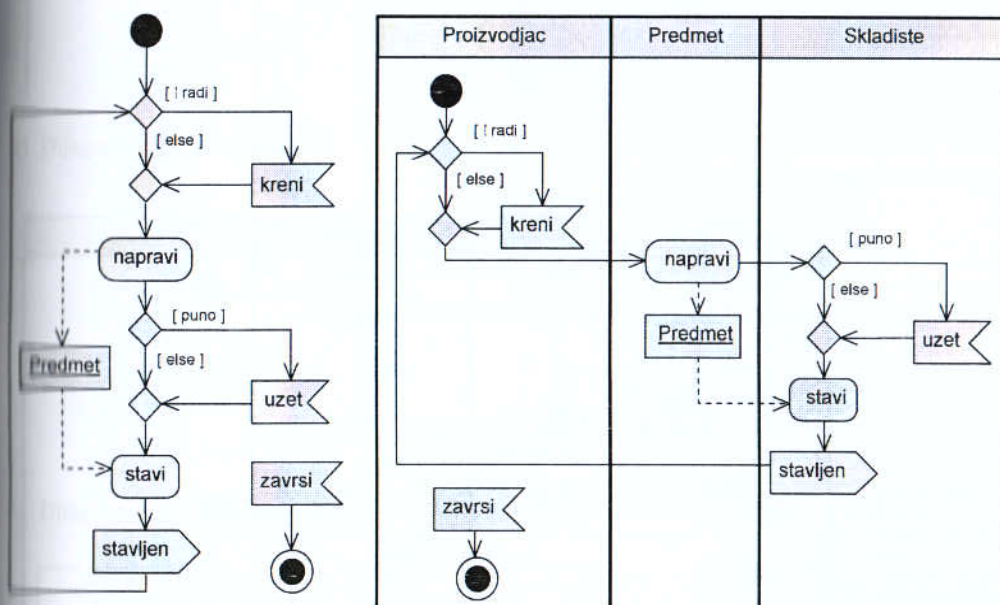
d) Dijagram komunikacije proizvođača



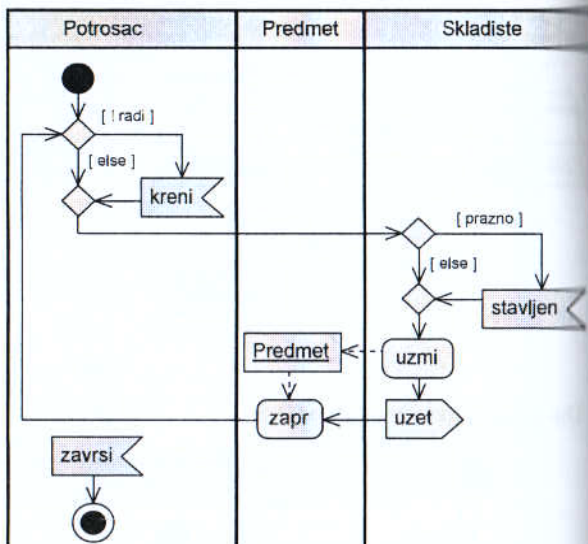
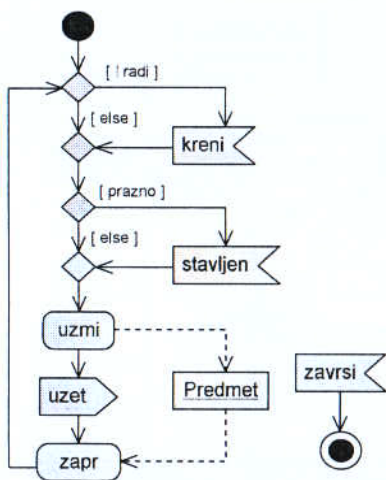
e) Dijagram komunikacije potrošača



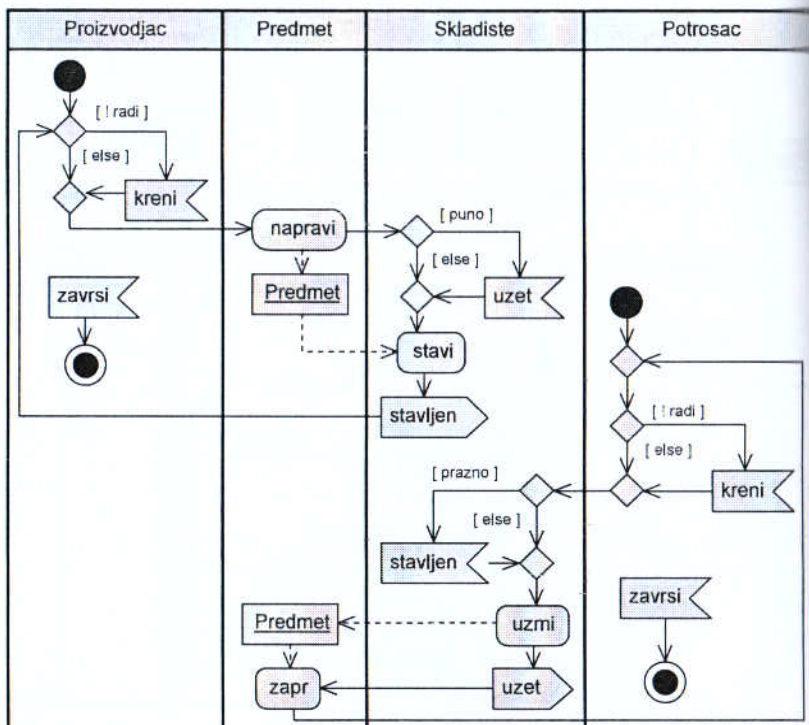
f) Dijagrami aktivnosti proizvođača



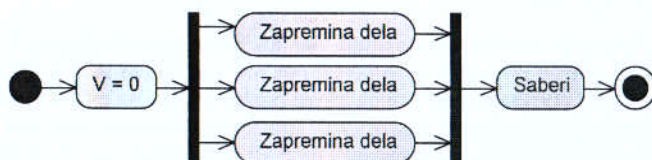
g) Dijagrami aktivnosti potrošača



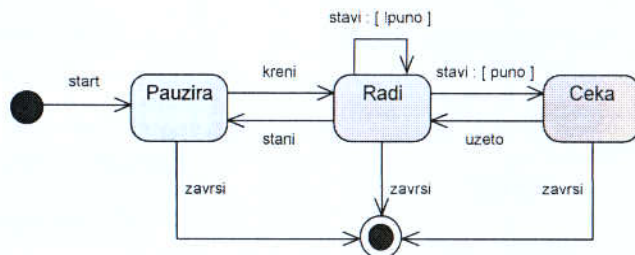
h) Zajednički dijagrami aktivnosti proizvođača i potrošača



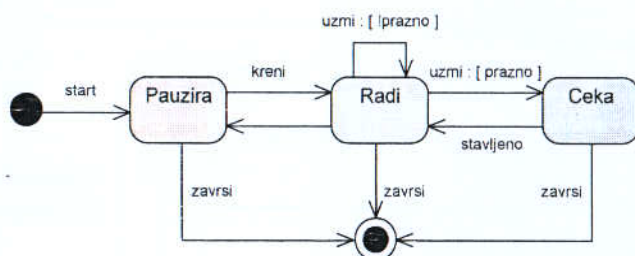
i) Dijagram aktivnosti izračunavanja zapremine sklopa



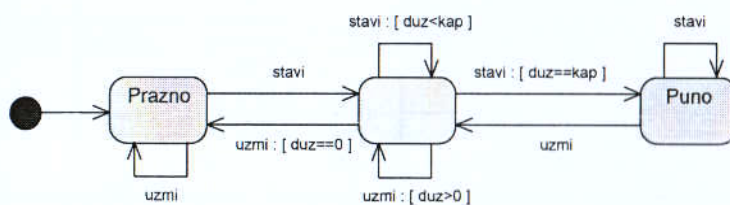
j) Dijagram stanja proizvođača



k) Dijagram stanja potrošača



l) Dijagram stanja skladišta



Zadatak 33 Predmet, skladište, pokazivač predmeta i nadzornici {K2, 15.12.2006.}

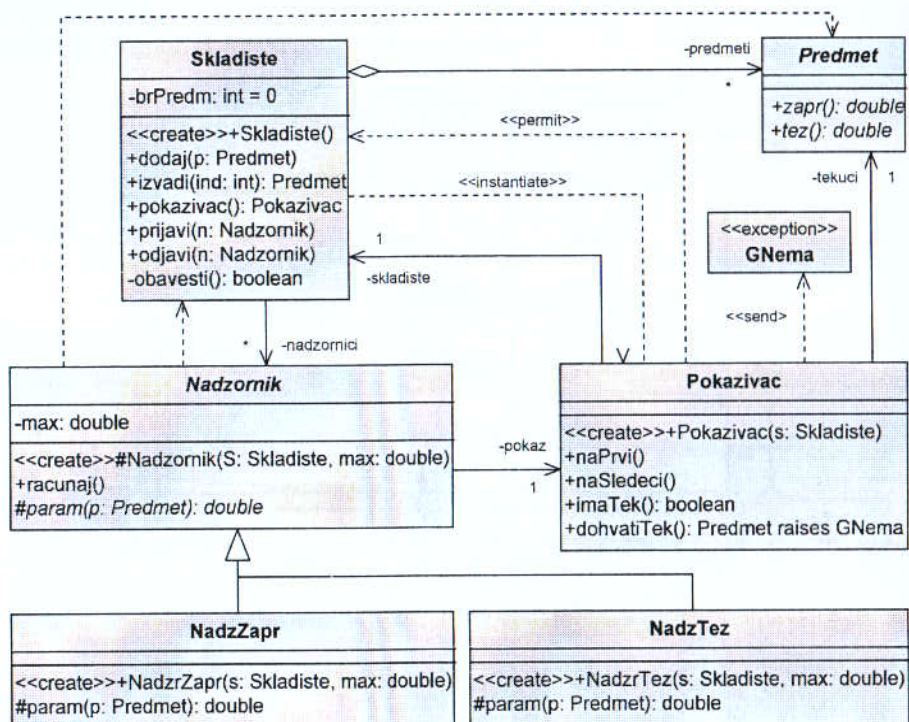
Apstraktnom predmetu može da se izračuna zapremina i težina. Skladište može da sadrži proizvoljan broj predmeta. Može da se doda jedan predmet i da se izvadi predmet sa zadatim rednim brojem (predmet se uklanja iz skladišta). Pomoću objekta pokazivača na tekući predmet skladišta mogu da se dohvataju uzastopni predmeti iz skladišta (predmeti ostaju u skladištu). Apstraktan nadzornik nadgleda određeni realni parametar skladišta. Parametar se određuje nakon svake promene u skladištu kao zbir odgovarajućih parametara svih predmeta u skladištu. Ukoliko je parametar veći od neke zadate veličine prekida se tekuća prozivka nadzornika i ispiše se poruka. Nadzornici mogu jedan po jedan da se prijavljuju i odjavljuju od skladišta. Nadzornik zapremine i nadzornik težine prate ukupnu zapreminu, odnosno težinu predmeta u skladištu.

Projektovati na jeziku UML prethodni sistem. Priložiti:

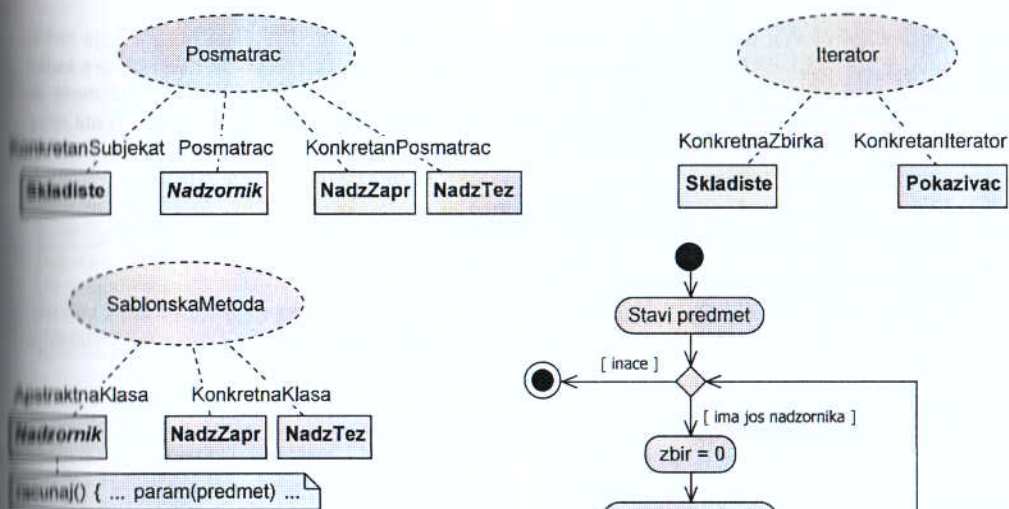
- dijagram klasa;
- prikaz korišćenih projektnih uzoraka;
- dijagram aktivnosti prilikom dodavanja jednog predmeta skladištu.

Rešenje:

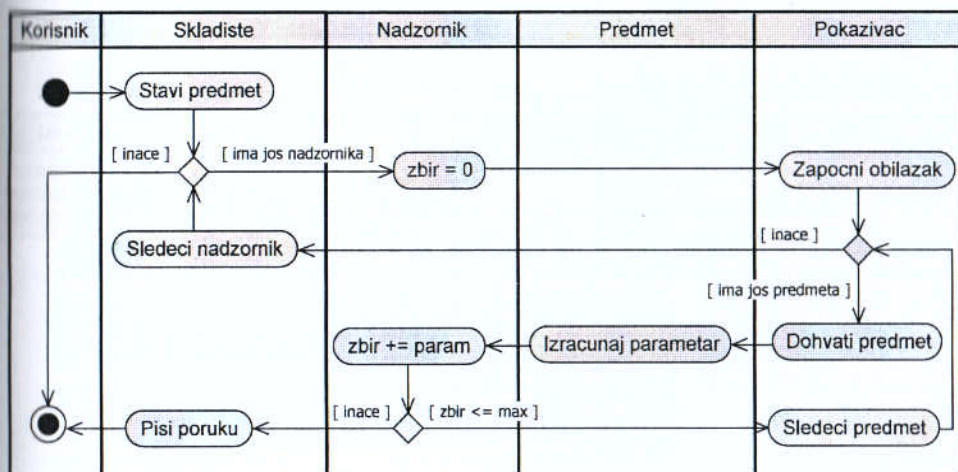
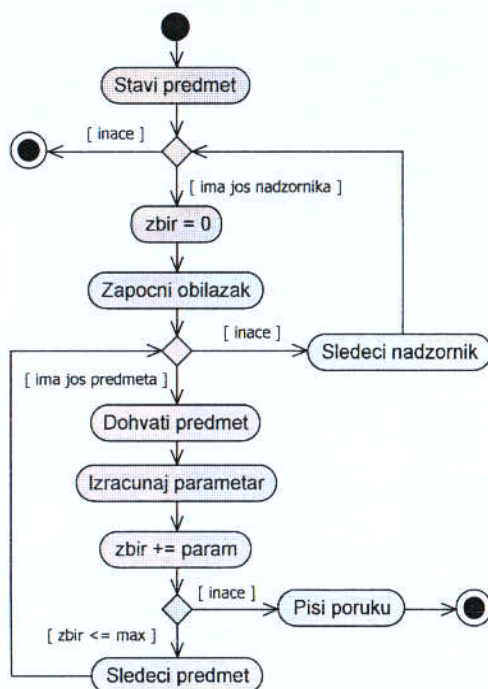
a) Dijagram klasa



b) Projektni uzorci



c) Dijagram aktivnosti



Zadatak 34 Roba, artikal, porez, popust, skladište, redosledi i izveštaj {K2, 06.12.2007.}

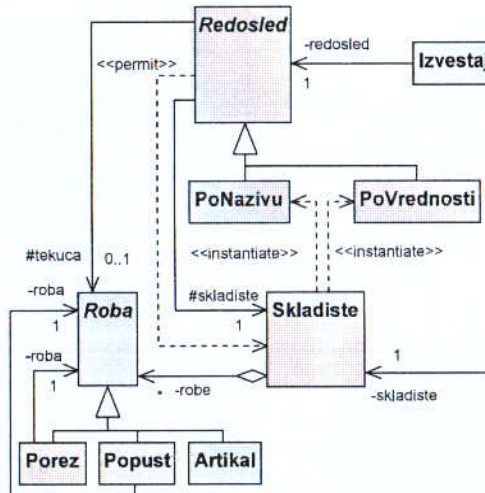
Roba predviđa dohvaćanje naziva, cene i količine i izračunavanje vrednosti. Artikal je roba koja sadrži naziv, jediničnu cenu i količinu. Roba s porezom je roba koja na cenu pridružene robe zaračunava zadati porez. Roba s popustom je roba koja od cene pridružene robe odbija zadati popust. Skladište može da sadrži proizvoljno mnogo robe. Stvara se prazno posle čega može da se doda i da se izvadi zadata roba. Robe u skladištu mogu da se dohvate po redosledu naziva i po redosledu vrednosti. Izveštaj može da ispisuje sadržaj zadatog skladišta. Može da se postavlja redosled prikazivanja robe u skladištu.

Projektovati na jeziku UML prethodni sistem. Priložiti:

- dijagram klasa;
- prikaz korišćenih projektnih uzoraka;
- dijagram objekata koji prikazuje skladište koje sadrži po jedan primerak robe bez poreza i popusta, samo s porezom, samo s popustom, s porezom posle čega se odbija popust i s popustom posle čega se dodaje porez;
- dijagrame sekvence i aktivnosti sastavljanja izveštaja za slučaj kada se na svaku robu prvo dodaje porez i posle toga odbija popust.

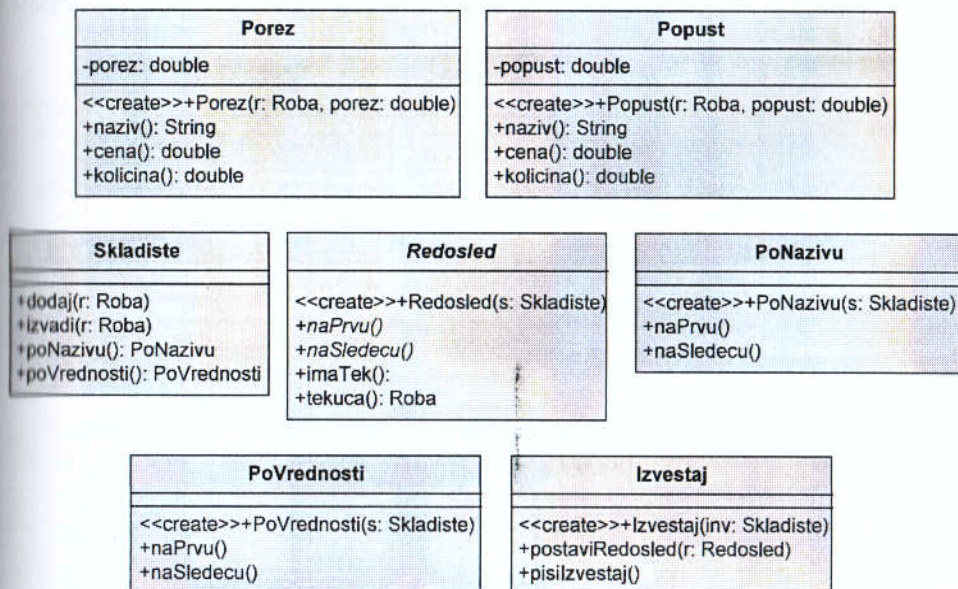
Rešenje:

a) Dijagram klasa

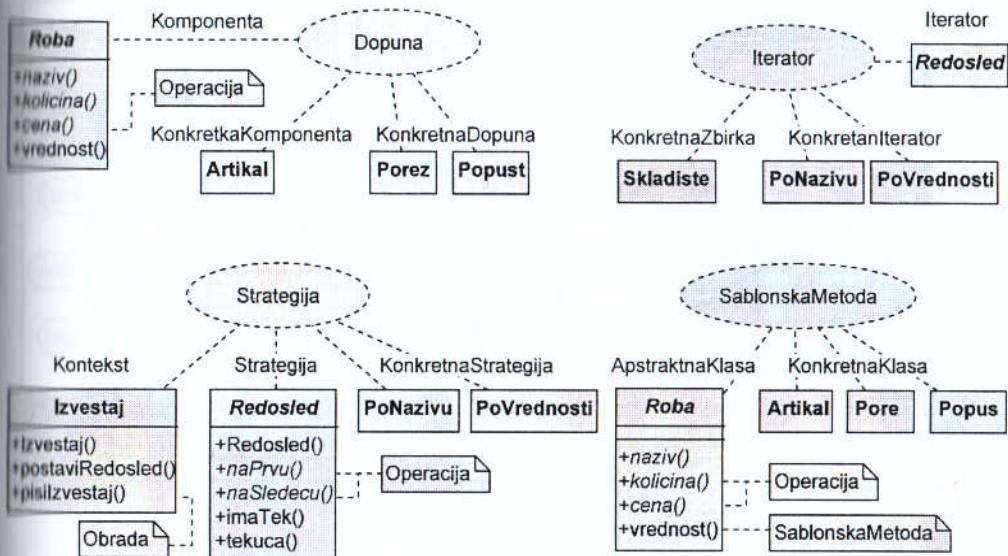


Roba
+naziv(): String +kolicina(): double +cena(): double +vrednost(): double

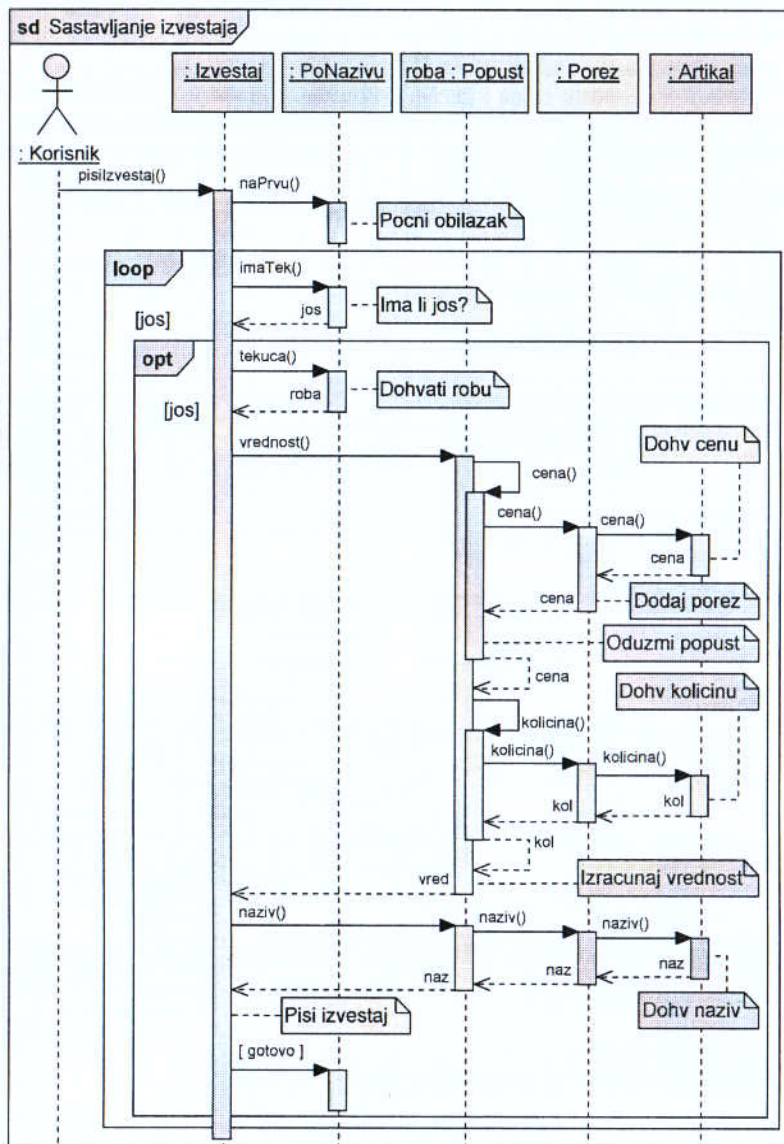
Artikal
-naziv: String -cena: double -kolicina: double <<create>>+Artikal(naziv: String, cena: double, kolicina: double) +naziv(): String +cena(): double +kolicina(): double



h) Projektni uzorci



d) Diagram sekvence



Zadatak 35 Vozila, mesto, parking, redosledi i semafor {K2, 04.12.2008.}

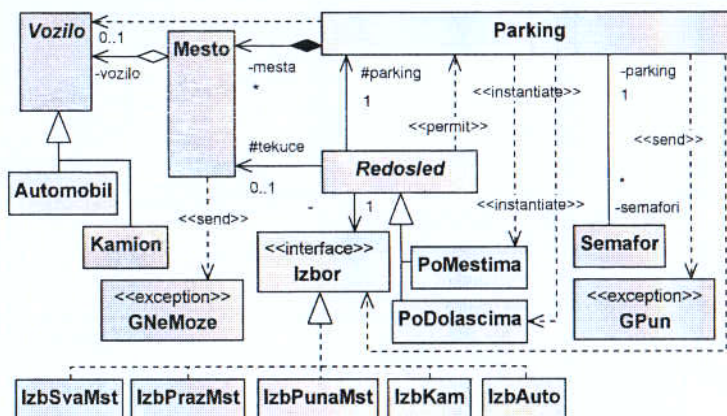
Vozilo ima zadatu širinu i dužinu koje mogu da se dohvate. Može da se odredi površina koju zauzima, da se dohvati vrsta vozila i da se sastavi tekstualni opis koji se sastoji od vrste i površine vozila. Automobil i kamion su vozila. Mesto ima zadatu širinu i dužinu. Može da se smesti jedno vozilo na mesto, da se vozilo dohvati ili ukloni i da se ispita da li je mesto prazno. Greška je ako vozilo ne može da se smesti na dato mesto. Parking ima određen broj mesta različite veličine. Stvara se prazan posle čega mesta mogu da se dodaju jedno po jedno. Na parking vozila mogu da dolaze i sa njega odlaze jedno po jedno. Greška je ako vozilo ne može da nađe slobodno mesto koje mu odgovara. Parking može da se obiđe po redosledu mesta i po redosledu dolazaka vozila a pri tome da se uzmu u obzir samo mesta sa vozilima jedne vrste, samo popunjena mesta, samo prazna mesta ili sva mesta. Postoji proizvoljan broj semafora koji prikazuju informacije o broju raspoloživih i zauzetih mesta na parkingu.

Projektovati na jeziku UML prethodni sistem. Priložiti:

- dijagram klasa;
- prikaz korišćenih projektnih uzoraka;
- dijagram aktivnosti pri dolasku jednog vozila na parking;
- dijagram stanja parkinga.

Rešenje:

a) Dijagram klasa



Vozilo
-sir: float -duz: float
<<create>>+Vozilo(s: float, d: float) +sirina(): float +duzina(): float +povrsina(): float +vrsta(): String +toString(): String

Automobil
<<create>>+Automobil(s: float, d: float) +vrsta(): String

Kamion
<<create>>+Kamion(s: float, d: float) +vrsta(): String

Mesto
-sir: float -duz: float -vremeDolaska: long
<<create>>+Mesto(s: float, d: float) +smesti(v: Vozilo, vreme: long) raises GNeMoze +prazno(): boolean +ukloni() +dohvati(): Vozilo +vremeDolaska(): long

Parking
+brojMesta(): int +brojVozila(): int +dodaj(m: Mesto) +dolazi(v: Vozilo, vreme: long) raises GPun +odlazi(v: Vozilo) +dodaj(s: Semafor) +izbaci(s: Semafor) -obavesti() +poDolascima(i: Izbor): PoDolascima +poMestima(i: Izbor): PoMestima

<<exception>> GNeMoze

<<exception>> GPun

Redosled
<<create>>+Redosled(p: Parking, i: Izbor) +naPrvo() +naSledece() +imaTek(): boolean +tekMst(): Mesto

PoDolascima

PoMestima

<<interface>> Izbor
+odgovara(v: Vozilo): boolean

IzbSvaMst

IzbPrazMst

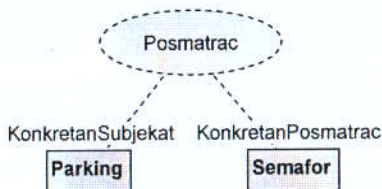
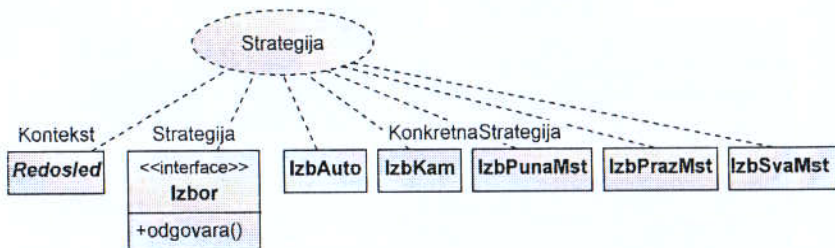
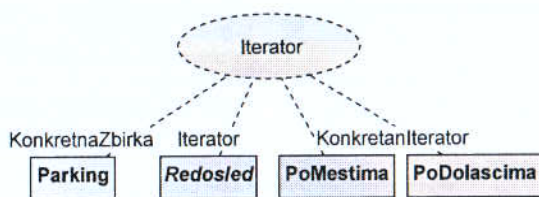
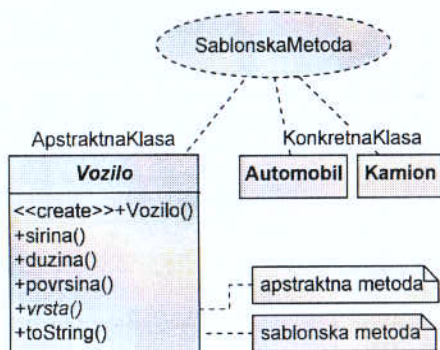
IzbPunaMst

IzbKam

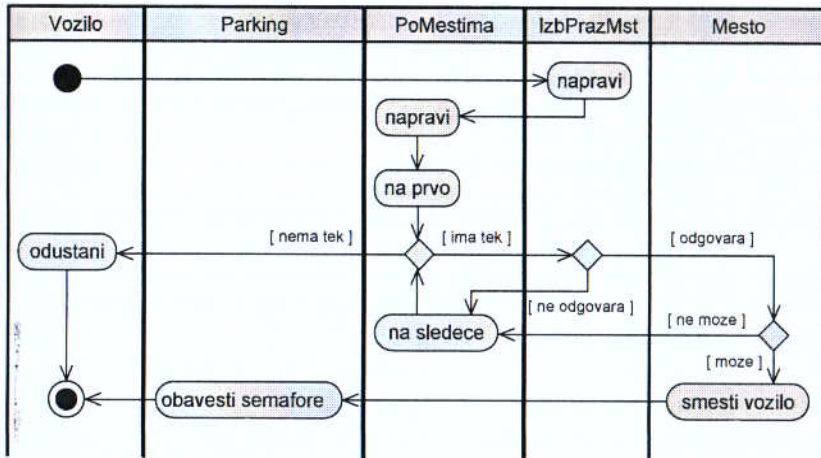
IzbAuto

Semafor
<<create>>+Semafor(p: Parking) +prikazi()

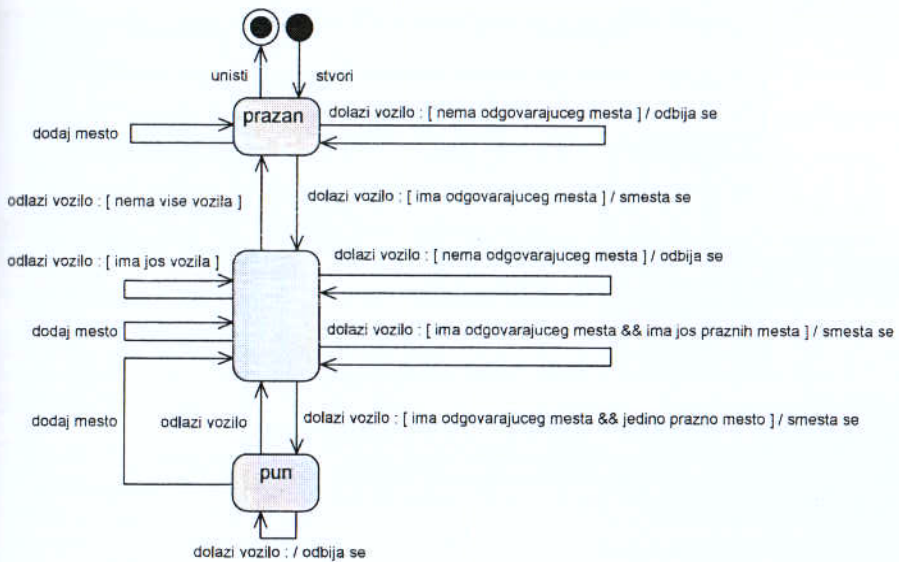
b) Projektni uzorci



g) Dijagram aktivnosti dolaska vozila



d) Dijagram stanja parkinga



Zadatak 36 Biblioteka, publikacije, dela i osobe {K2, 04.12.2009.}

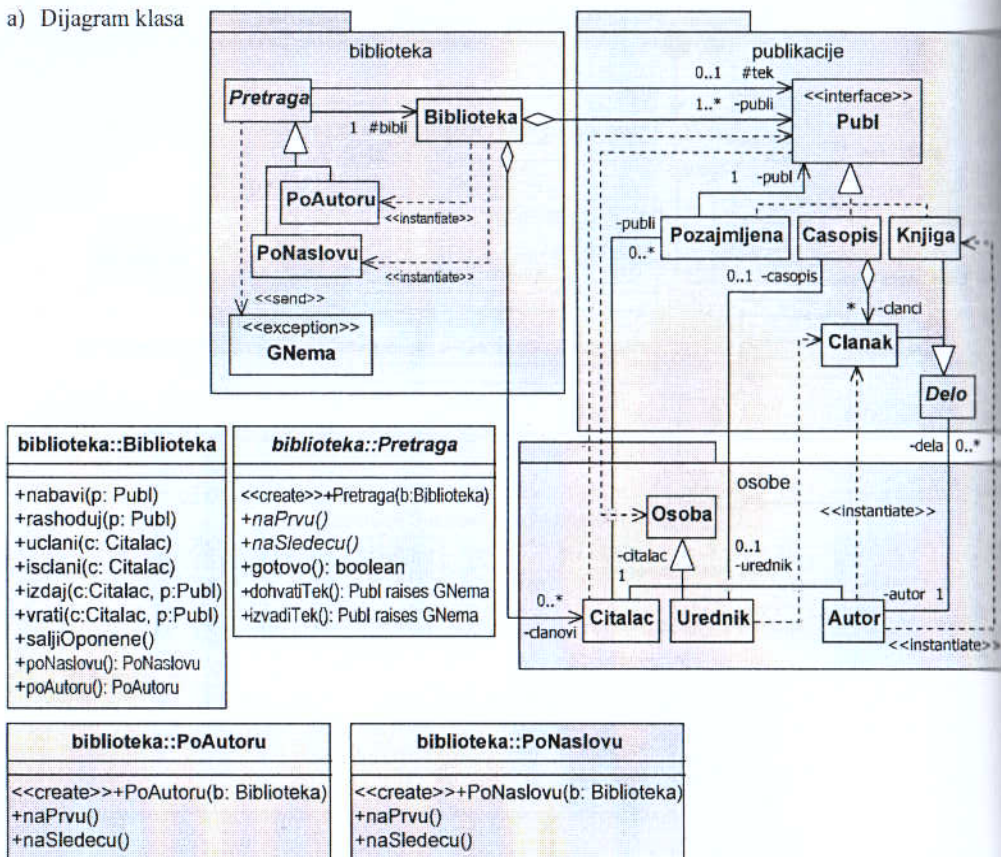
Biblioteka može da sadrži proizvoljan broj publikacija. Može da nabavlja i da rashoduje zadatu publikaciju, da učlani i išclani zadatog čitaoca, da izda na određeni rok i primi nazad zadatu publikaciju od zadatog čitaoca i da pošalje opomenu svim čitaocima koji su prekoračili rok za vraćanje publikacije. Fond biblioteke može da se pretražuje po naslovima i po autorima. Publikaciji može da se dohvati naslov i osoba koja može biti autor ili urednik. Knjiga i časopis su publikacije. Pozajmljena publikacija je publikacija koja predstavlja publikaciju izdatu zadatom čitaocu. Sadrži datum pozajmljivanja i rok za vraćanje publikacije. Časopis ima urednika koji može da se promeni. Časopis se stvara prazan posle čega se članci dodaju pojedinačno. Delo ima autora (samo jednog), naslov i tekstualni sadržaj. Knjiga i članak su dela. Osoba ima ime. Čitalac je osoba koja može da pozajmi i da vrati zadatu publikaciju u zadatu biblioteku i da primi opomenu za zadatu nevratu publikaciju. Urednik je osoba koja može da se zaduži da uređuje zadati časopis. Može da primi članak za objavljivanje. Autor je osoba koja može da napiše jednu knjigu ili članak.

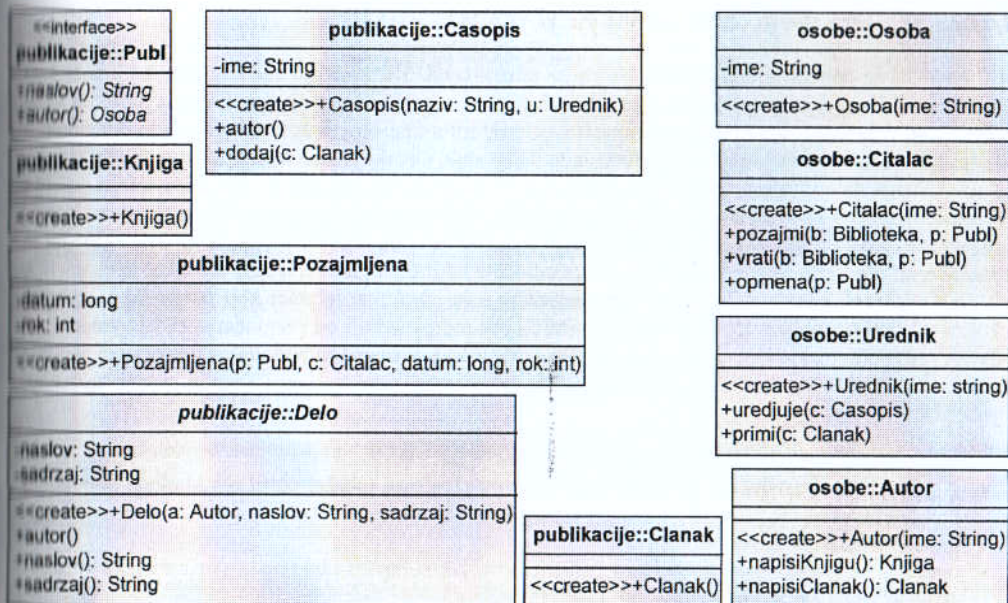
Projektovati na jeziku UML prethodni sistem. Priložiti:

- dijagram klasa;
- prikaz korišćenih projektnih uzoraka;
- dijagram stanja kroz koje prolazi časopis.

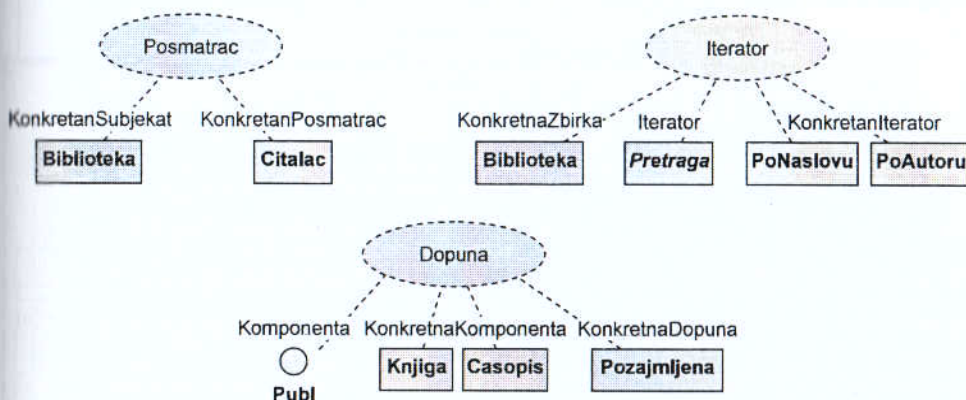
Rešenje:

a) Dijagram klasa

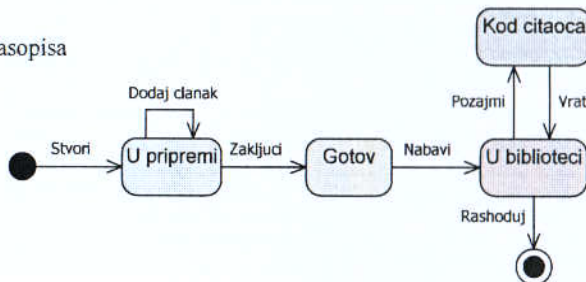




b) Projektni uzorci



e) Dijagram stanja časopisa



Zadatak 37 Dela, galerija, osobe i časovnik {K2, 07.12.2010.}

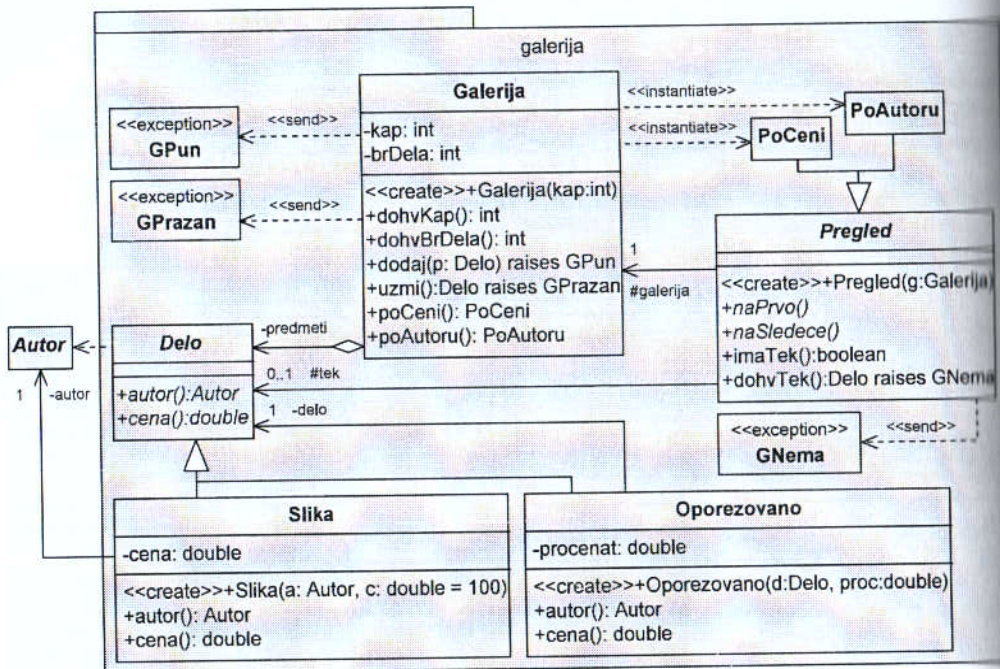
Delu može da se dohvati njegov autor i da mu se odredi cena. Slika je delo zadatog autora i ima zadatu cenu. Oporezovano delo je delo na čiju cenu se zaračunava zadati procenat poreza. Galerija može da sadrži zadat broj dela. Može da se dohvati kapacitet galerije i trenutni broj dela u galeriji, da se doda delo u galeriju i da se uzme delo iz galerije po redosledu dodavanja. Greška je ako se pokuša staviti delo u punu galeriju ili uzeti delo iz prazne galerije. Sadržaj galerije može da se pregleda po redosledu imena autora i po redosledu cena dela. Osoba ima zadato ime koje može da se dohvati i u slučajnim vremenskim intervalima, između zadatog najdužeg i najkraćeg trajanja, izvršava neku radnju. Može da joj se saopšti da je protekao neki vremenski period na osnovu čega odlučuje da li treba da izvrši radnju. Autor je osoba čija se radnja sastoji od stvaranja dela i stavljanja u zadatu galeriju. Slikar je autor koji stvara slike u tehnici akvarela, ulja ili bakroreza. Kolekcionar je osoba čija se radnja sastoji od pregledanja zadate galerije po zadatom redosledu i kupovine jednog od dela u galeriji. Aktivan časovnik periodično obaveštava sve prijavljene osobe da je protekao zadati vremenski interval. Osobe se prijavljuju i odjavljuju kod časovnika pojedinačno.

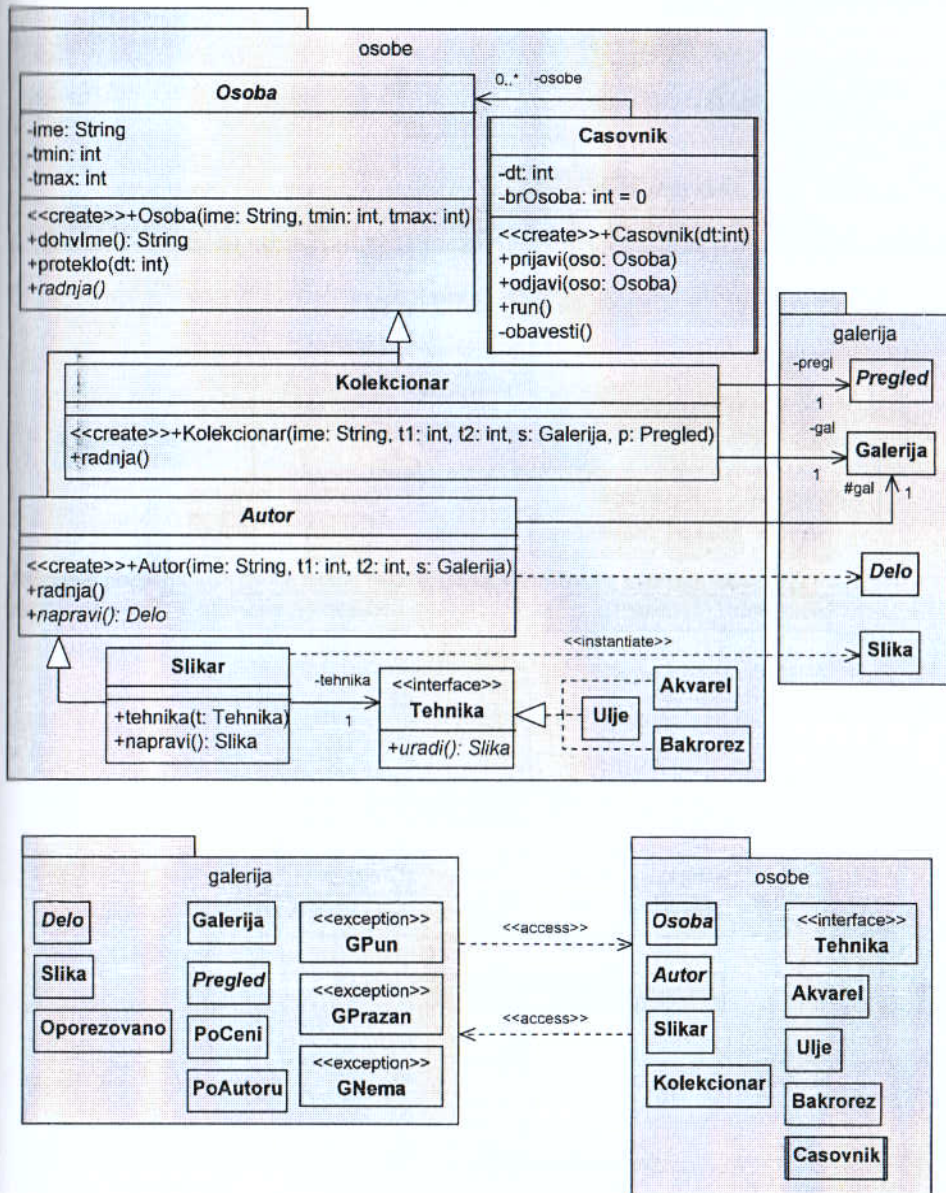
Projektovati na jeziku UML prethodni sistem. Priložiti:

- dijagram klasa razvrstanih u pakete (odnose među klasama i sadržaj pojedinih klasa na potrebnom broju slika);
- prikaz korišćenih projektnih uzoraka;
- dijagram aktivnosti rada časovnika i podaktivnosti koju obavi osoba nakon svakog isteka vremenskog intervala časovnika.

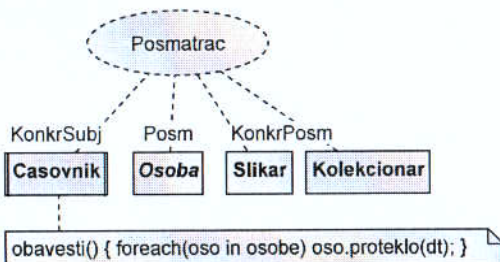
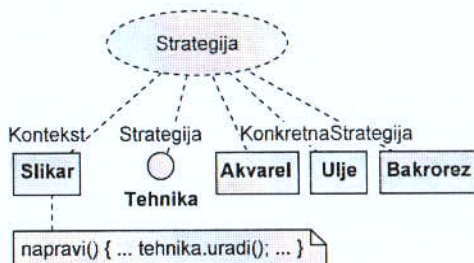
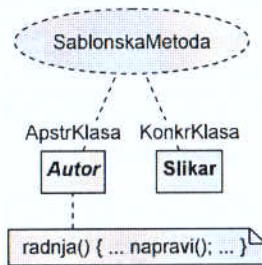
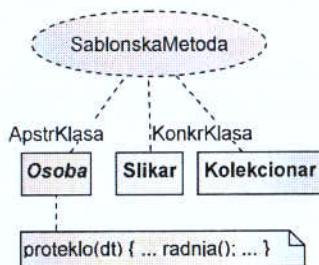
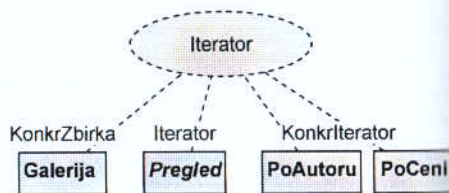
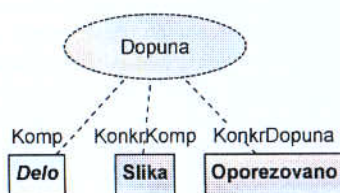
Rešenje:

a) Dijagrami klasa i paketa

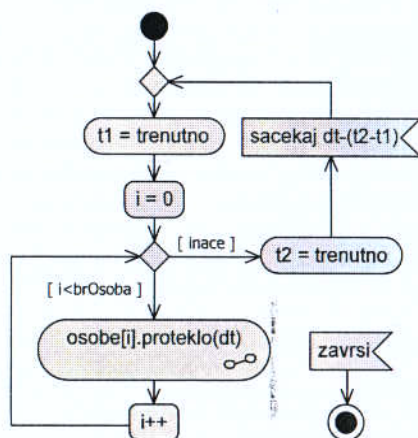




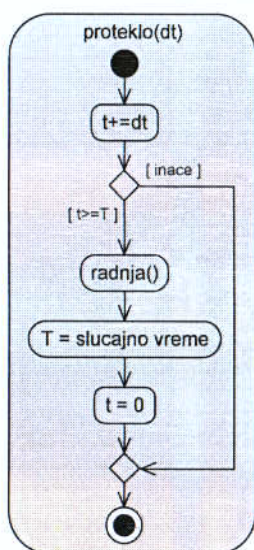
b) Projektni uzorci



i) Dijagram aktivnosti rada časovnika



ii) Dijagram podaktivnosti osobe nakon svakog isteka vremenskog intervala časovnika



Zadatak 38 Akvizicija podataka {K2, 29.11.2011.}

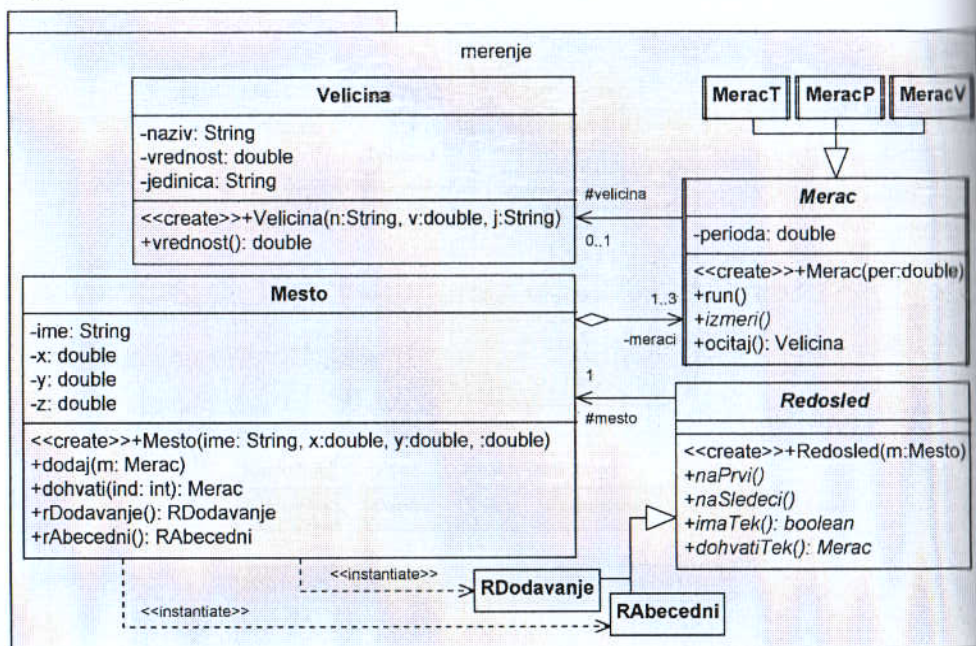
Fizička veličina ima naziv, vrednost i jedinicu mere. Aktivni merač može da izmeri i zapamti vrednost fizičke veličine do očitavanja. On vrši merenje zadatom periodom. Rezultat očitavanja merača je fizička veličina. Merači temperature, pritiska i vlage mere odgovarajuće fizičke veličine. Merno mesto je opisan imenom i prostornim koordinatama. Opremljeno je meračima koji mu se pojedinačno dodaju (najviše 3). Može se dohvatiti pojedini merač. Merači na mernom mestu mogu da se očitavaju različitim redosledom i to prema redosledu dodavanja ili prema abecednom redosledu fizičkih veličina. Rezultat merenja na mernom mestu sadrži ime mernog mesta i zbirku fizičkih veličina koje se dodaju pojedinačno. Aktivni računar ciklički vrši neku obradu. Radna stanica je računar koji poseduje sanduče za odlaganje proizvoljnog broja rezultata merenja. Radna stanica dohvata rezultate merenja iz sandučeta radi prikazivanja fizičkih veličina, pri čemu ih sanduče isporučuje ili po redosledu prispeća, ili najpre poslednji prispeli. Kontroler je računar koji ima zadato merno mesto i redosled očitavanja merača. On očitava sve merače na datom mernom mestu, otkriva promenu fizičke veličine i formira rezultat merenja. Kontroleru mogu da se pridruže radne stanice na kojima se prikazuju promene fizičkih veličina na mernom mestu. Kontroler šalje radnoj stanici rezultate merenja samo kada postoji promena neke fizičke veličine, a radna stanica ih smešta u svoje poštansko sanduče radi kasnijeg prikazivanja.

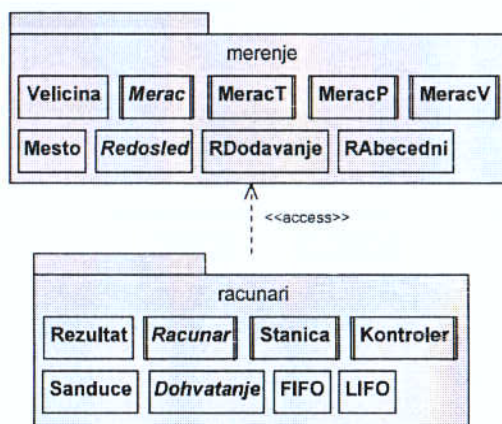
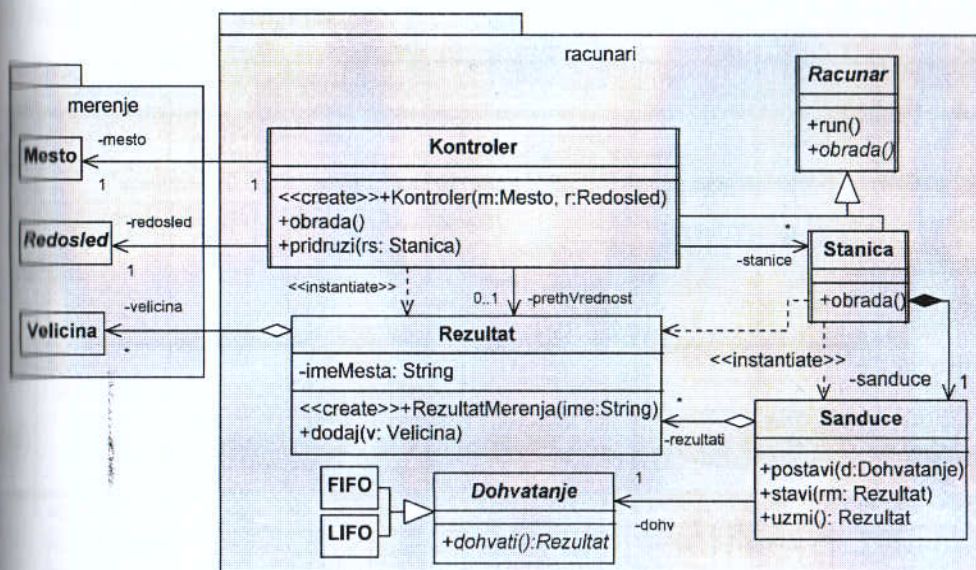
Projektovati na jeziku UML prethodni sistem. Priložiti:

- dijagram klasa razvrstanih u pakete (odnose među klasama i sadržaj pojedinih klasa na potrebnom broju slika);
- prikaz korištenih projektnih uzoraka;
- dijagram aktivnosti koji koristeći plivačke staze prikazuje tok ponavljajućih aktivnosti od merenja veličine meraćem do njenog prikaza na radnoj stanici.

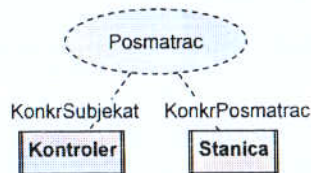
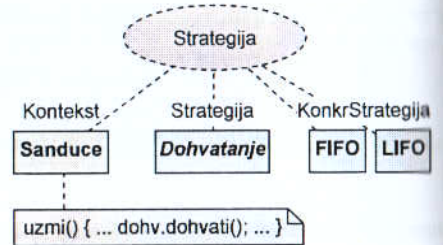
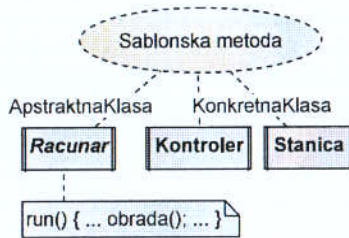
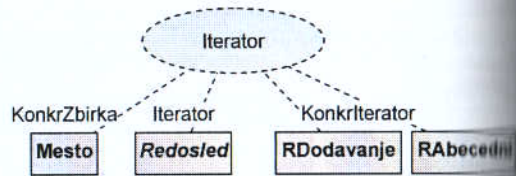
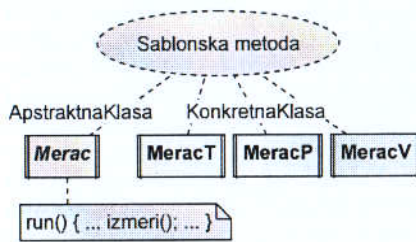
Rešenje:

a) Dijagrami klasa i paketa

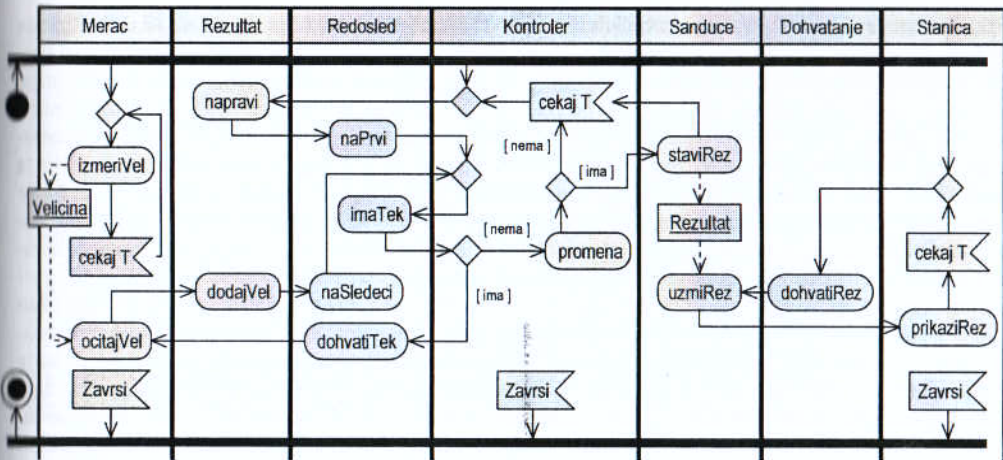




b) Projektni uzorci



e) Dijagram aktivnosti merenja veličine



Zadatak 39 Distribucija električne energije {K2, 27.11.2012.}

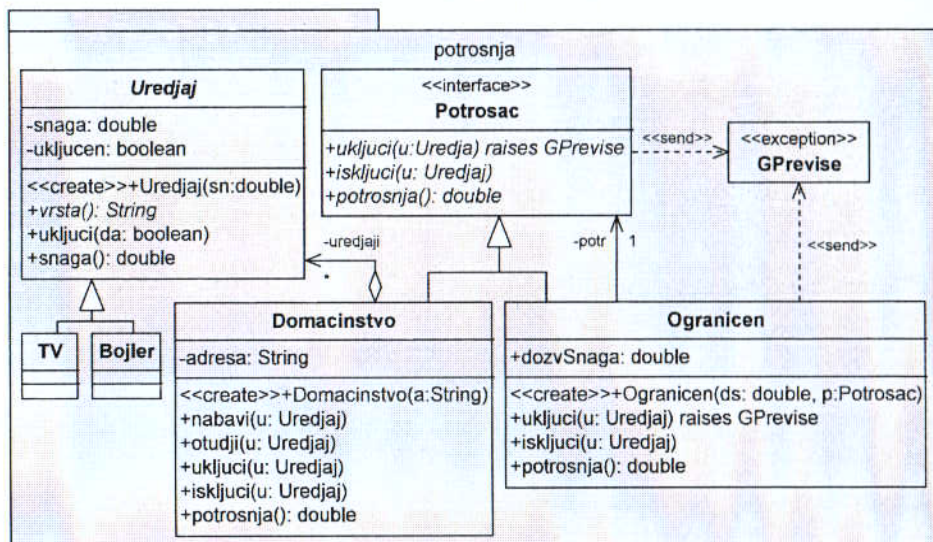
Električni uređaj ima zadatu snagu potrošnje i može biti uključen ili isključen. Može da se dohvati vrsta uređaja, da se uključi/isključi i da se dohvati njegova trenutna snaga potrošnje. Televizor i bojler su uređaji. Potrošač predviđa uključivanje i isključivanje zadatog uređaja i određivanje trenutne ukupne snage potrošnje svih svojih uređaja. Domaćinstvo je potrošač koji ima zadatu adresu. Može da nabavi, otuđi, uključi i isključi zadati uređaj. Ograničeni potrošač ima zadatu dozvoljenu maksimalnu trenutnu potrošnju svih uređaja pridruženog potrošača. Greška je ako bi se pri uključivanju nekog uređaja prekoračila ta granica. Elektrana ima naziv, maksimalnu snagu koju može da razvija i trenutnu razvijenu snagu. Može da se dohvati maksimalna i trenutna snaga, oznaka vrste elektrane i da se promeni trenutno razvijena snaga za zadati procenat. Greška je ako bi se elektrana preopteretila. Postoje termo, hidro i nuklearne elektrane. Na aktivnu distribuciju električne energije može da se poveže i odveže data elektrana i da se priključi i isključi dati potrošač. Može da se odredi ukupna maksimalna i trenutna snaga priključenih elektrana kao i ukupna potrošnja priključenih potrošača. Distribucija ciklički proverava da li je trenutna potrošnja unutar zadatog maksimalnog i minimalnog procenta u odnosu na trenutnu razvijenu snagu elektrana. Ako je ukupna potrošnja prevelika, nalaže elektranama da povećaju, a ako je premala, da smanje proizvodnju električne energije. Elektrane mogu da se obilaze po redosledu maksimalne razvijene snage, trenutno razvijene snage ili trenutnog procentualnog iskorišćenja kapaciteta. Moguće je obilaziti samo termoelektrane, samo hidroelektrane, samo nuklearne elektrane ili sve elektrane.

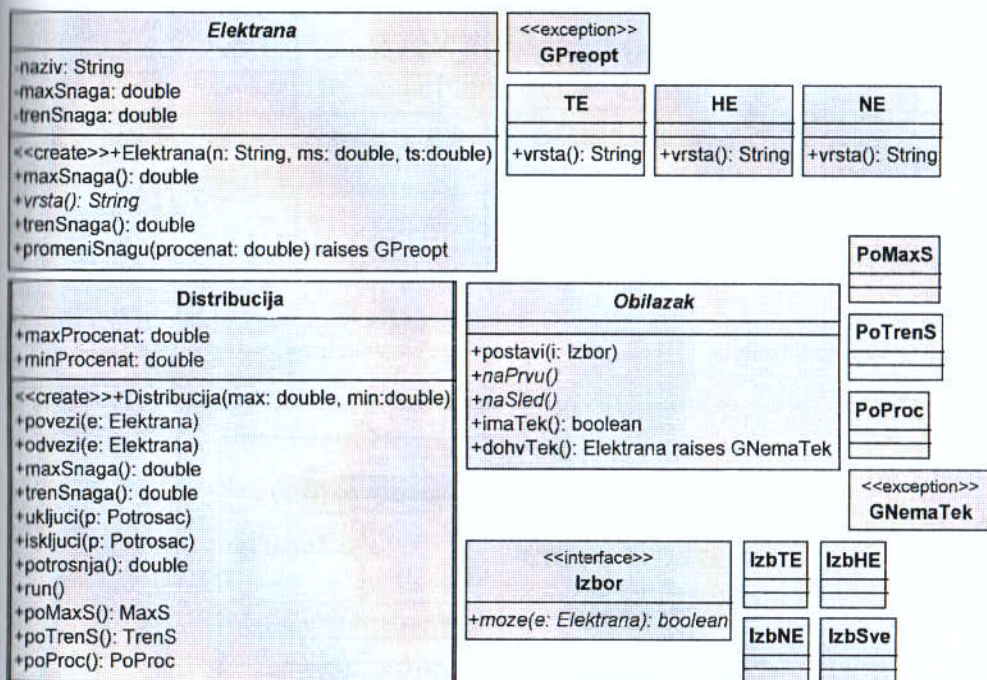
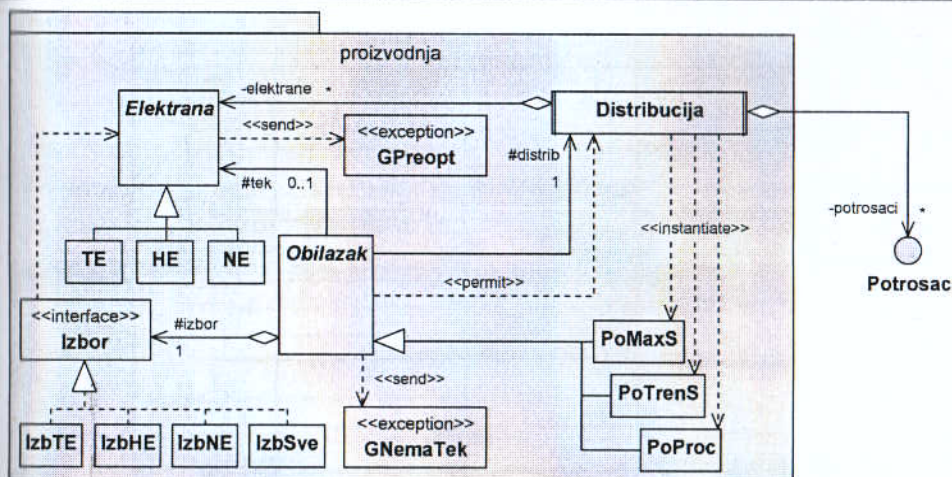
Projektovati na jeziku UML prethodni sistem. Priložiti:

- dijagram klasa razvrstanih u pakete (odnose među klasama i sadržaj pojedinih klasa na potrebnom broju slika);
- prikaz korišćenih projektnih uzoraka;
- dijagram aktivnosti koji prikazuje izračunavanje ukupne razvijene snage hidroelektrana;
- dijagram stanja jednog uređaja od nabavke do otuđenja.

Rešenje:

a) Dijagrami klasa

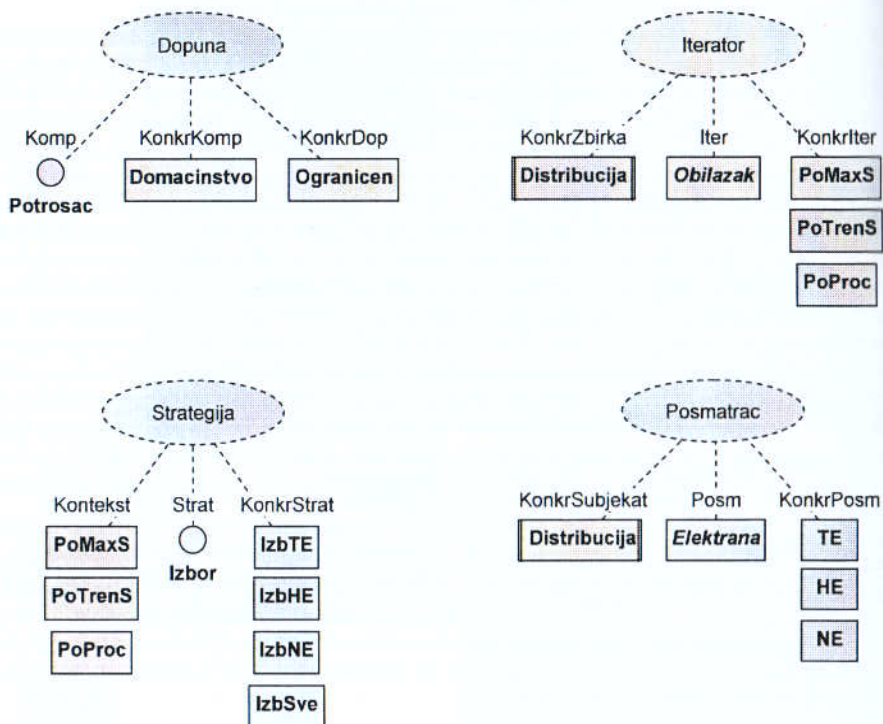




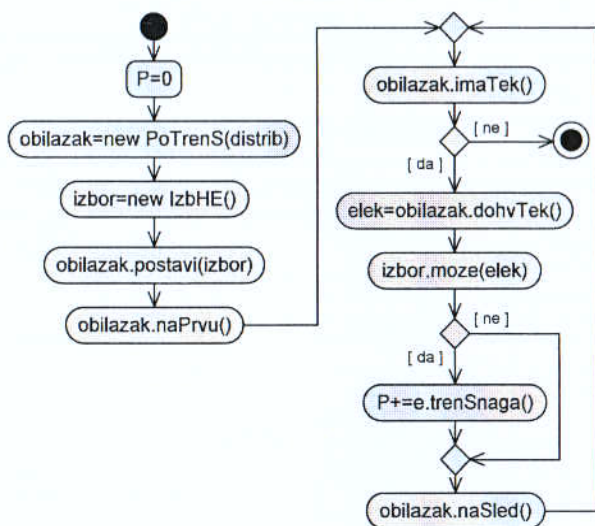
b) Dijagram paketa

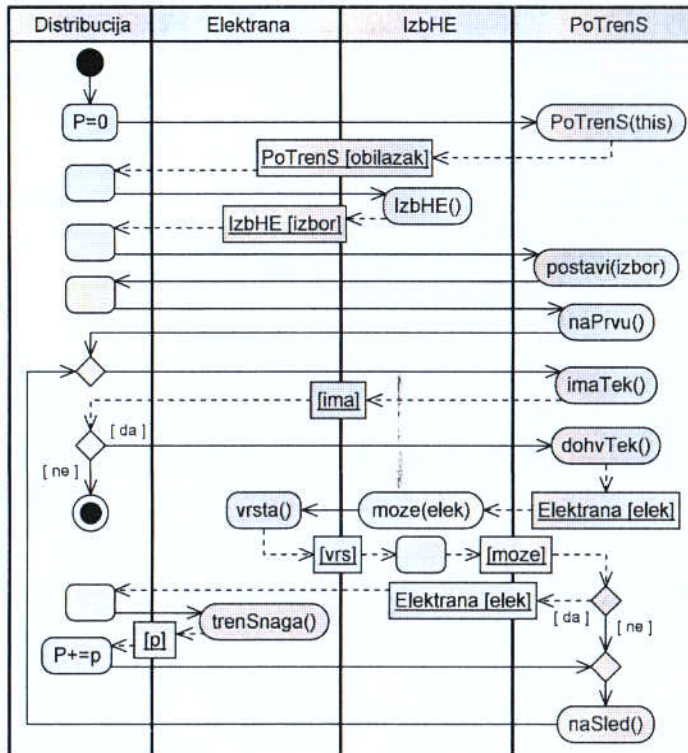


c) Projektni uzorci

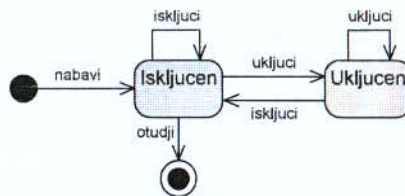


d) Dijagram aktivnosti izračunavanja ukupne razvijene snage hidroelektrana





e) Dijagram stanja potrošača

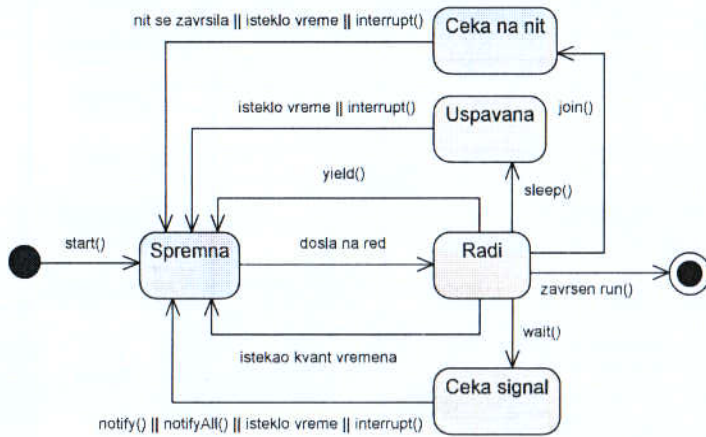


Zadatak 40 Dijagram stanja i aktivnosti niti u jeziku Java

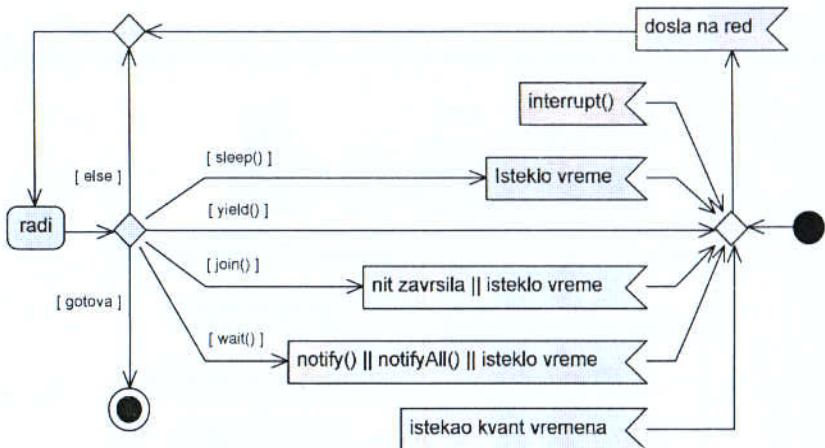
Nacrtati dijagram stanja i dijagram aktivnosti rada niti u jeziku *Java*.

Rešenje:

a) Dijagram stanja

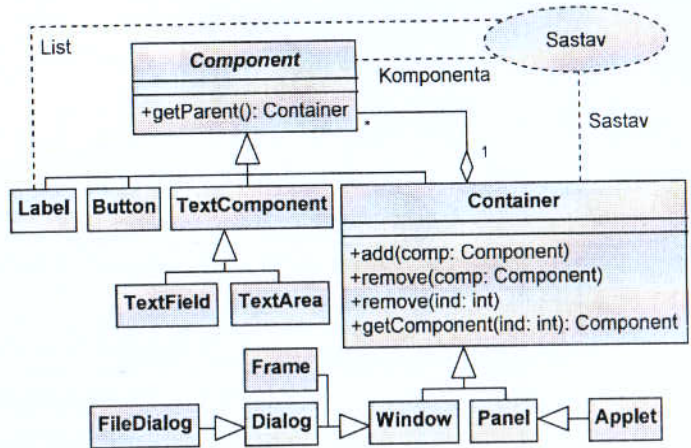
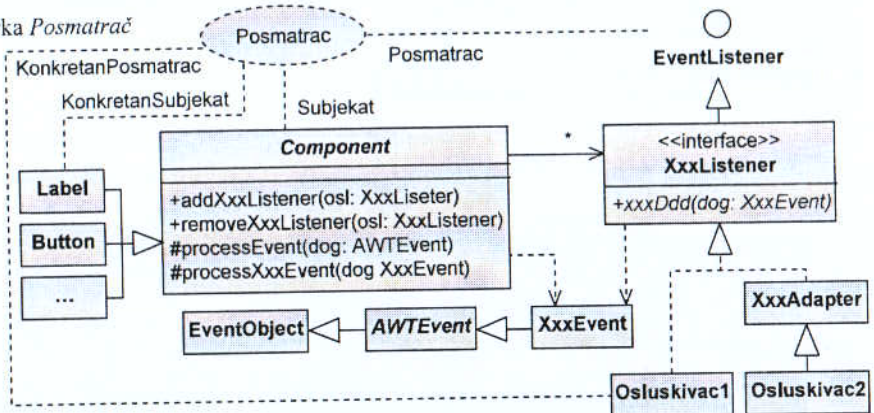
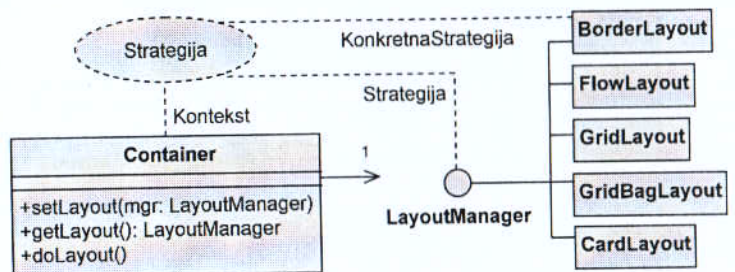


b) Dijagram aktivnosti



Zadatak 41 Projektni uzorci u paketu AWT

Naći primere projektnih uzoraka u paketu AWT jezika *Java*.

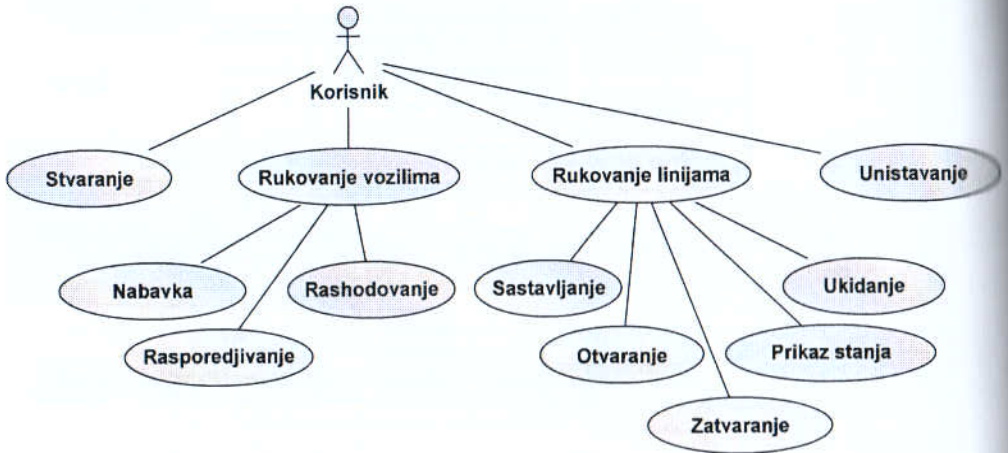
Rešenje:a) Primer uzorka *Sastav*b) Primer uzorka *Posmatrač*c) Primer uzorka *Strategija*

Zadatak 42 Gradski saobraćaj (projektni uzorak *Stanje*)

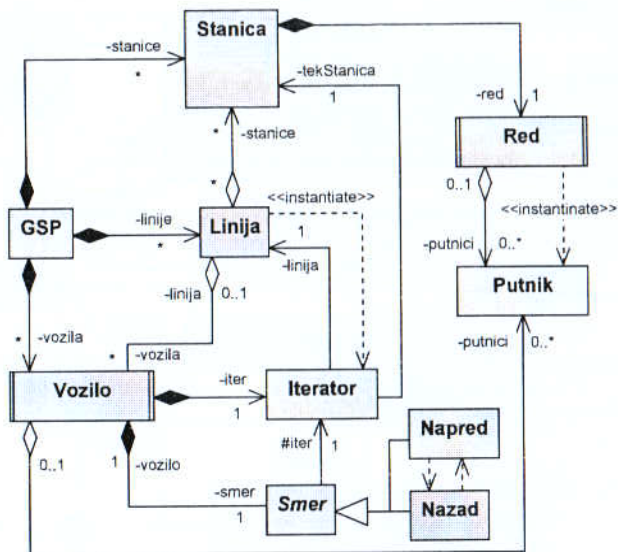
Projektovati na jeziku *UML* model sistema za simuliranje rada gradskog saobraćaja.

Rešenje:

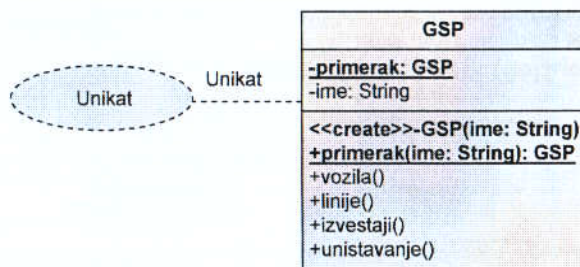
a) Dijagram slučajeva korišćenja



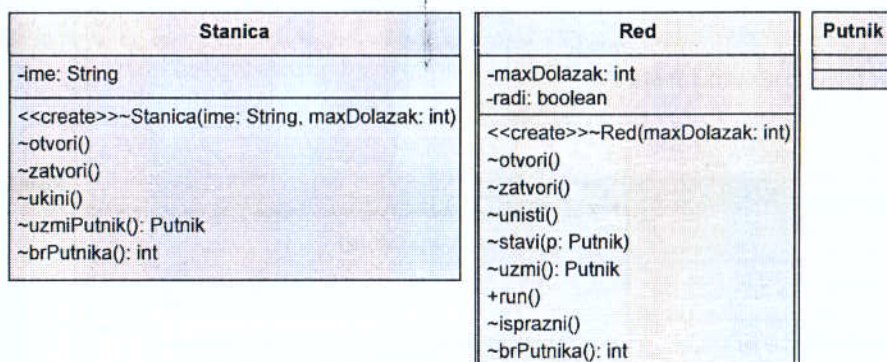
b) Dijagram klasa



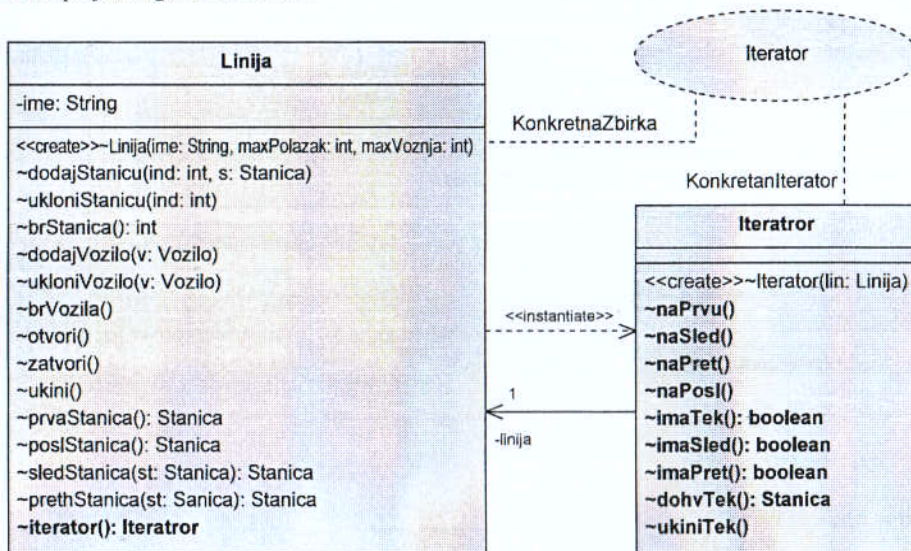
i) Primer projektnog uzorka *Unikat*



d) Klase *Stanica*, *Red* i *Putnik*

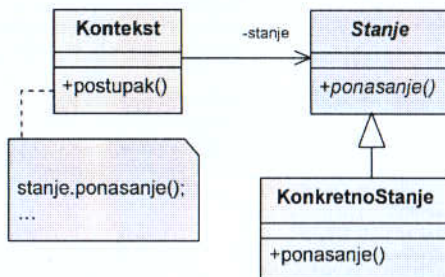
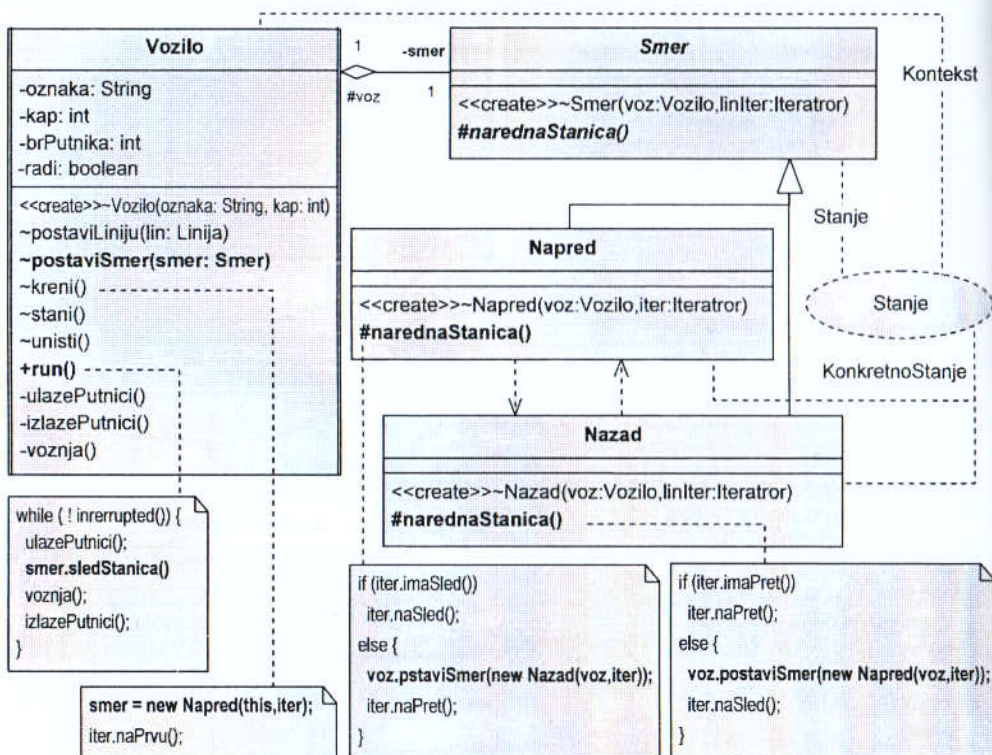


e) Primer projektnog uzorka *Iterator*

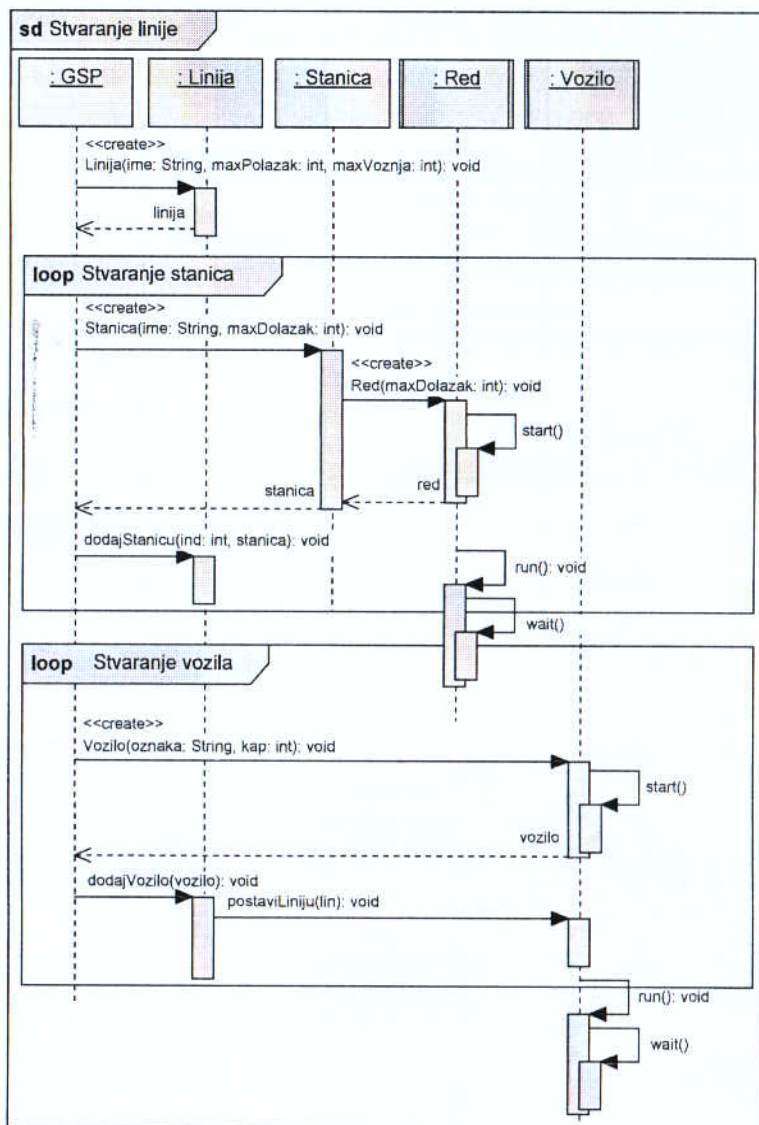


f) Projektni uzorak **Stanje** (*State*)

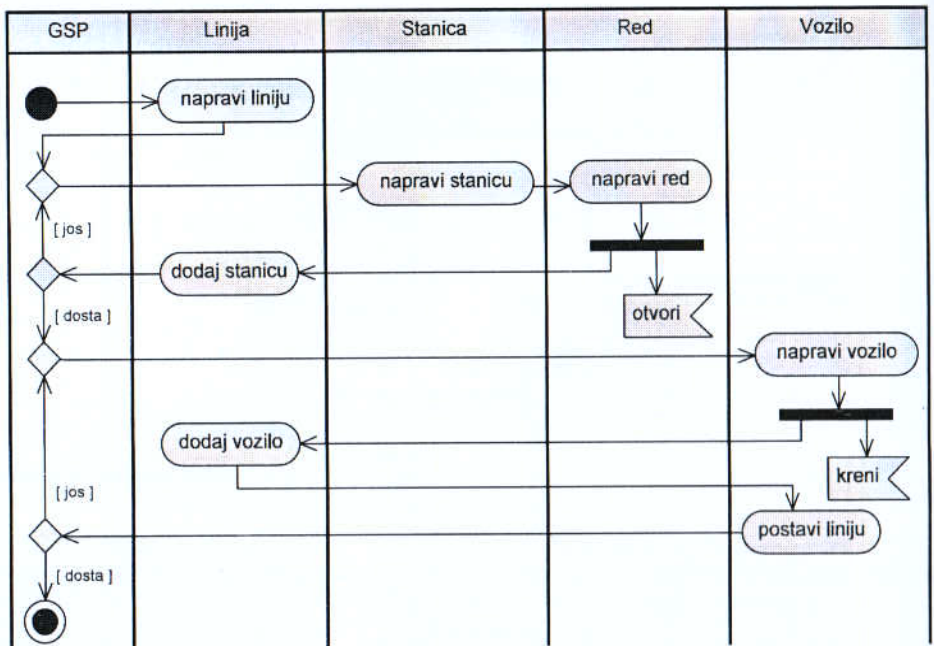
- objektni uzorak ponašanja
- menjanje ponašanja objekta kad se promeni njegovo unutrašnje stanje
 - kao da je objekat promenio svoju klasu
- kontekst obavlja postupak koji zavisi od stanja konteksta

g) Primer projektnog uzorka *Stanje*

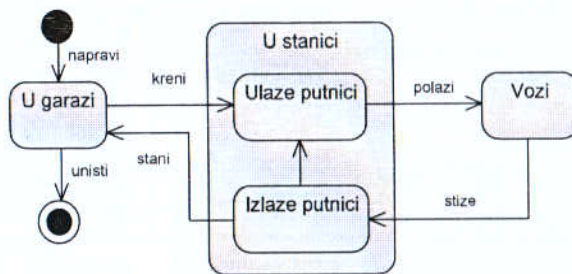
h) Dijagram sekvence sastavljanja linije



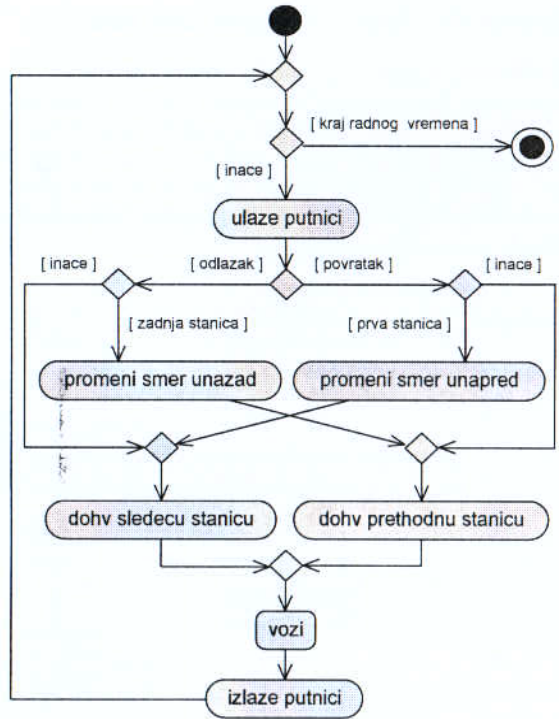
i) Dijagram aktivnosti sastavljanja linije



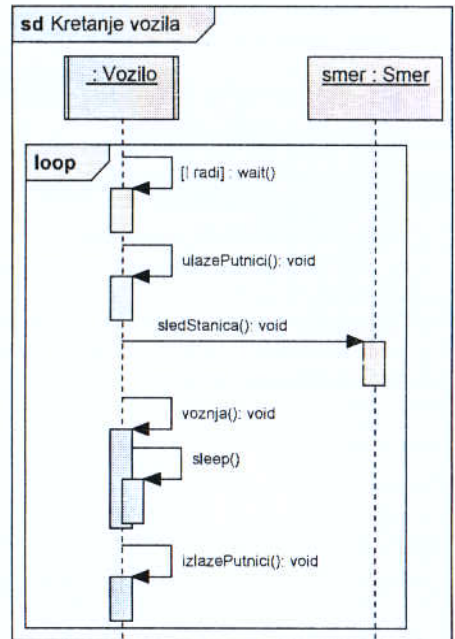
j) Dijagram stanja vozila



k) Dijagram aktivnosti kretanja vozila



l) Dijagram sekvence kretanja vozila



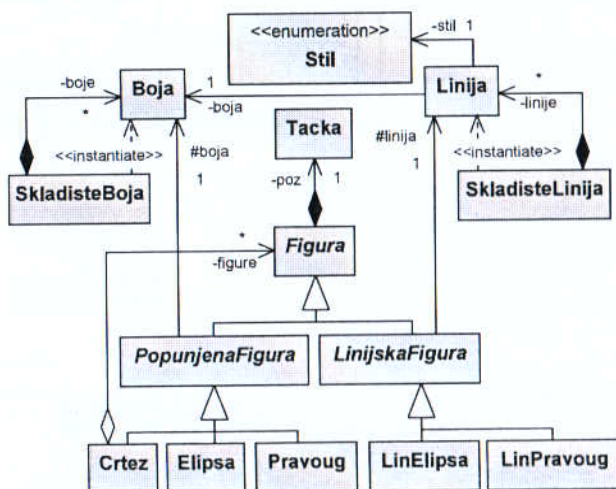
Zadatak 43 Figure u ravni (projektni uzorak *Muva*)

Projektovati na jeziku *UML* model sledećeg problema:

Apstraktna figura može da se crta na grafičkoj korisničkoj površi unutar pravougaonog prostora zadatih dimenzija sa gornjim levim uglom u zadatoj tački. Postoje popunjene figure koje se popunjavaju zadatom bojom i linijske figure koje se crtaju linijom zadatog stila (puna, isprekidana, tačkasta i crta-tačka), debljine i boje. Pravougaonici i elipse mogu da se crtaju popunjeno i linijski. Crtež je popunjena figura koja može da sadrži proizvoljan broj figura proizvoljne vrste. Pretpostaviti da se tipično koristi mali broj različitih boja i različitih vrsta linija.

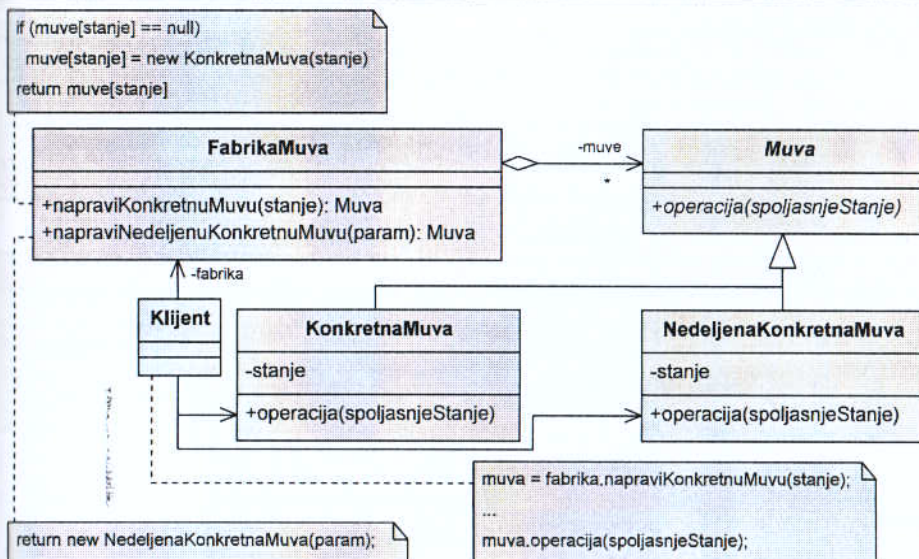
Rešenje:

a) Dijagram klasa

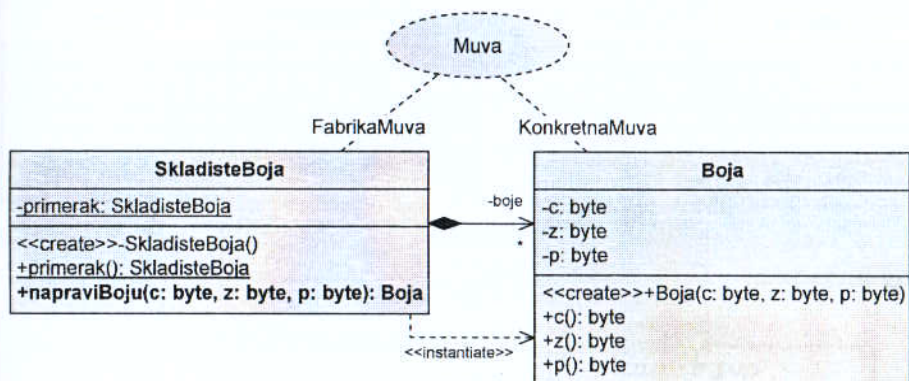


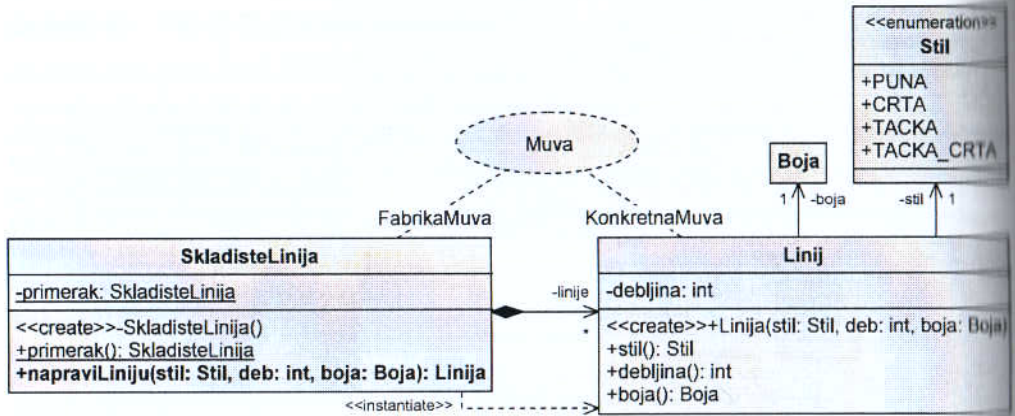
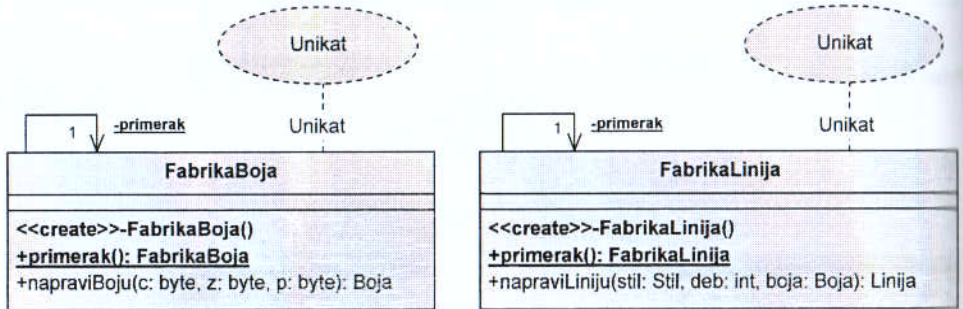
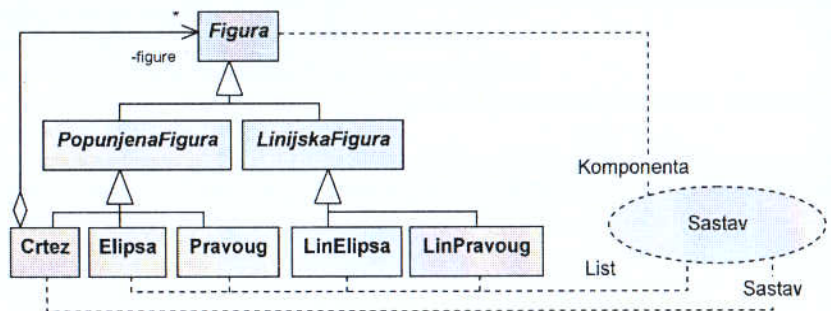
b) Projektni uzorak **Muva** (*Flyweight*)

- objektni uzorak strukture
- deljenje objekata radi izbegavanja hiperprodukcije objekata
 - objekti ili nemaju stanja ili im stanje ne zavisi od konteksta korišćenja (ne menja se posle stvaranja)
- klijent stvara objekte isključivo posredstvom fabrike objekata koja vodi računa o tome da ne postoje dva objekta s istim *unutrašnjim* stanjem, a pri korišćenju klijent navodi *spoljašnje* stanje zavisno od konteksta korišćenja



e) Primeri uzorka Muva



d) Primeri uzorka *Unikat*e) Primer uzorka *Sastav*

Tacka	Figura
-x: int -y: int	#sir: int #vis: int
<<create>>+Tacka(x: int, y: int) +x(): int +y(): int	<<create>>#Figura(poz: Tacka, s: int, v: int) +poz(): Tacka +sir(): int +vis(): int +crtaj()

LinijaskaFigura
<<create>>#LinijaskaFigura(poz: Tacka, a: int, b: int, lin: Linija) +linija(): Linija

LinPravoug
<<create>>+linPravoug(poz: Tacka, s: int, v: int, lin: Linija) +crtaj()

LinElipsa
<<create>>+LinElipsa(poz:Tacka, s:int, v:int, lin:Linija) +crtaj()

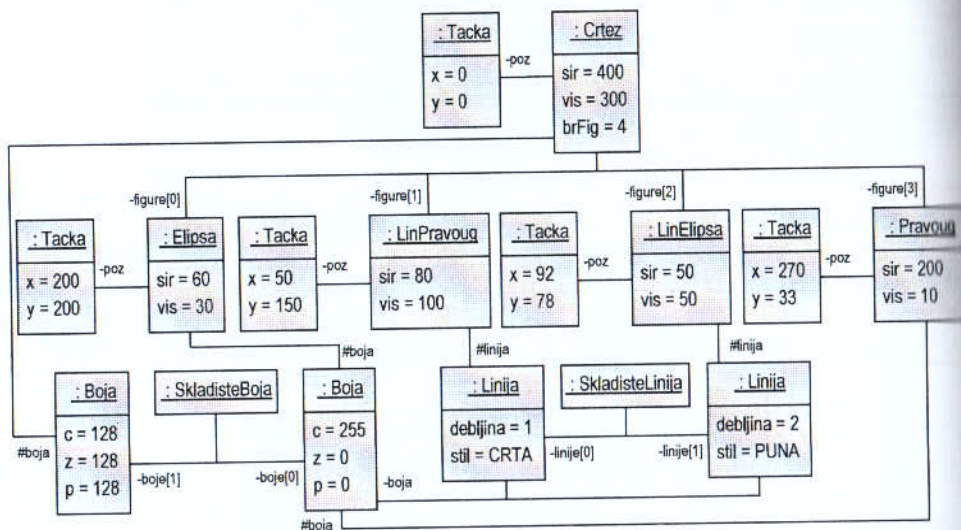
PopunjenaFigura
<<create>>#PopunjenaFigura(poz: Tacka, a: int, b: int, boja: Boja) +boja(): Boja

Elipsa
<<create>>+Elipsa(poz: Tacka, s: int, v: int boja:Boja) +crtaj()

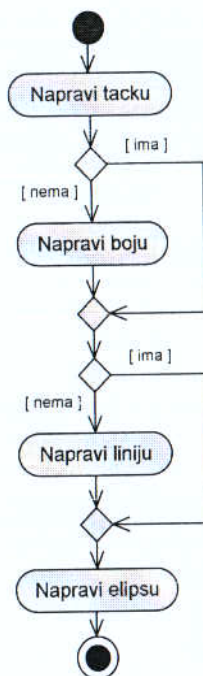
Pravoug
<<create>>+Pravoug(poz: Tacka, s: int, v:int, boja: Boja) +crtaj()

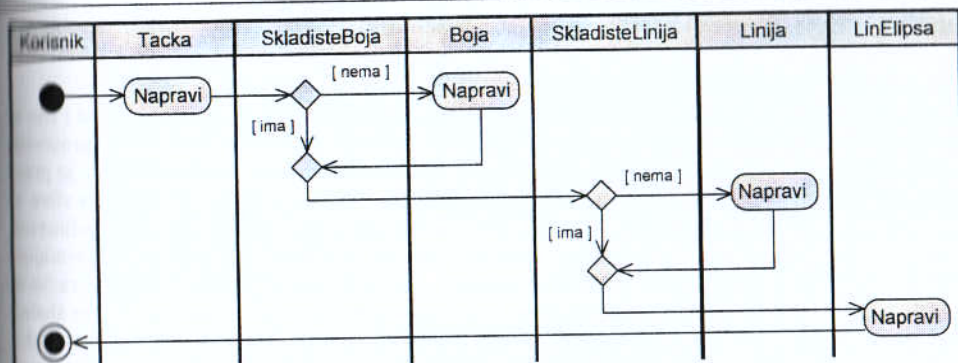
Crtez
+brFig: int
<<create>>+Crtez(poz: Tacka, s: int, v: int, boja: Boja) +crtaj() +dodaj(fig: Figura) +ukloni(fig: Figura) +ukloni(ind: int)

f) Dijagram objekata

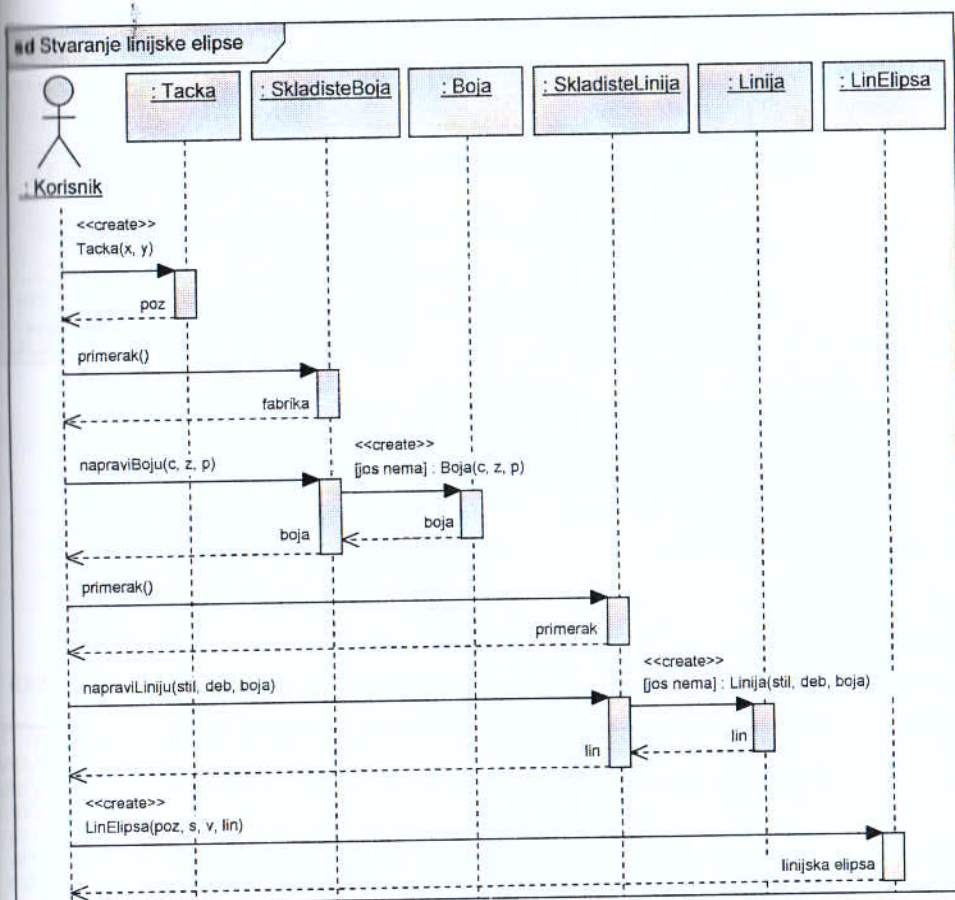


g) Dijagram aktivnosti stvaranja linijske elipse





h) Dijagram sekvence stvaranja linijske elipse



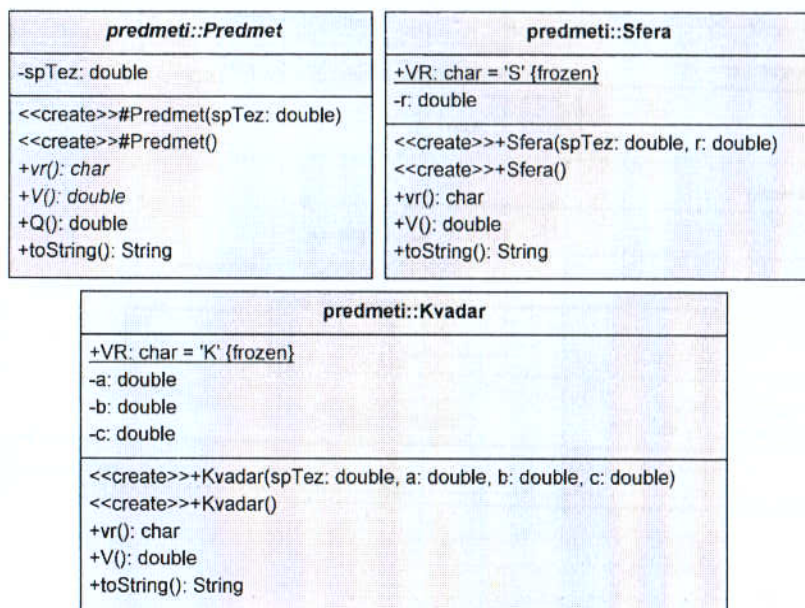
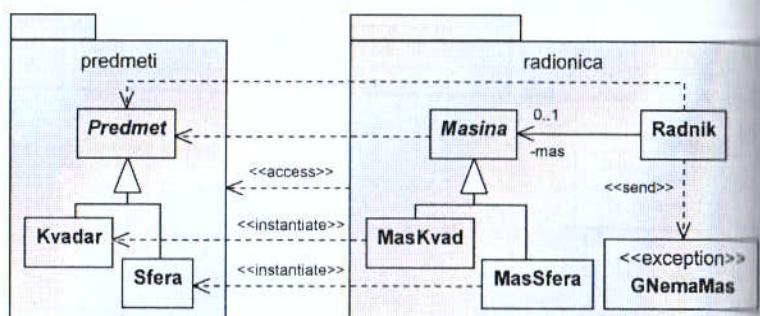
Zadatak 44 Predmeti, mašine i radnik (projektni uzorak *Proizvodna metoda*)

Projektovati na jeziku *UML* model sledećeg problema:

Apstraktna mašina može da proizvodi apstraktan predmet od materijala zadate specifične težine i zna se koliko je predmeta od datog materijala proizvela ta mašina. Može da se dohvati oznaka vrste proizvoda koje data mašina proizvodi i broj proizvedenih predmeta. Može da se promeni materijal od čega se prave predmeti i da se dohvati specifična težina korišćenog materijala. Mašina za kvadre i mašina za sfere su mašine koje mogu da proizvode kvadre, odnosno sfere zadatih dimenzija. Radnik ima ime i jedinstven, automatski generisan celobrojan identifikator. Izrađuje predmete određene vrste pomoću odgovarajuće mašine. Mašina na kojoj radnik radi može da se promeni. Zna se koliko je predmeta radnik izradio na mašini na kojoj trenutno radi. Radnik može da ne bude raspoređen ni na jednu mašinu. U tom slučaju pokušaj izrade predmeta ili dohvatanja broja izrađenih predmeta je greška. Tekstualni opis radnika sadrži ime i identifikator radnika. U slučaju da je radnik raspoređen na neku mašinu dodaje se i vrsta predmeta koje trenutno izrađuje i broj proizvoda koje je izradio na toj mašini od početka rada na njoj.

Rešenje:

a) Dijagrami klasa



radionica::Masina	radionica::MasSfera
#spTez: double #napravila: int	-r: double
<<create>>#Masina(s: double) +vrsta(): char +napravila(): int +dohvSpTez(): double +postaviSpTez(s: double) +napravi(): Predmet	<<create>>+MasSfera(spTez: double, r: double) +r(): double +vrsta(): char +napravi(): Sfera

radionica::MasKvad
-a: double -b: double -c: double
<<create>>+MasKvad(spTez: double, a: double, b: double, c: double) +a(): double +b(): double +c(): double +vrsta(): char +napravi(): Kvadar

radionica::Radnik	<<exception>> radionica::GNemaMas
-posId: int -id: int = ++posId -ime: String -napravio: int	-ime: String -id: int
<<create>>+Radnik(ime: String) +id(): int +ime(): String +rasporedi(m: Masina): Radnik +dohvatiMas(): Masina +napravio(): int raises GNemaMas +napravi(): Predmet raises GNemaMas +toString(): String	<<create>>+GNemaMas(ime: String, id: int) +ime(): String +id(): int +toString(): String

b) Kôd na jeziku *Java* koji je generisao *StarUML*

```
// @ File Name : Predmet.java
package predmeti;
public abstract class Predmet {
    private double spTez;
    protected void Predmet(double spTez) {}
    protected void Predmet() {}
    public abstract char vr();
    public abstract double V();
    public final double Q() {}
    public String toString() {}
}
```

Potvrđena je opcija `IsLeaf`.


```
// @ File Name : Sfera.java
package predmeti;
public class Sfera extends Predmet {
    public static final char VR = 'S';
    private double r;
    public void Sfera(double spTez, double r) {}
    public void Sfera() {}
    public char vr() {}
    public double V() {}
    public String toString() {}
    public char vr() {}
    public double V() {}
}
```

← Odabrano je: Changeability=FROZEN

```
// @ File Name : Masina.java
package radionica;
import predmeti.*;
public abstract class Masina {
    protected double spTez;
    protected int napravila;
    protected void Masina(double s) {}
    public abstract char vrsta();
    public int napravila() {}
    public double dohvSpTez() {}
    public void postaviSpTez(double s) {}
    public abstract Predmet napravi();
}
```

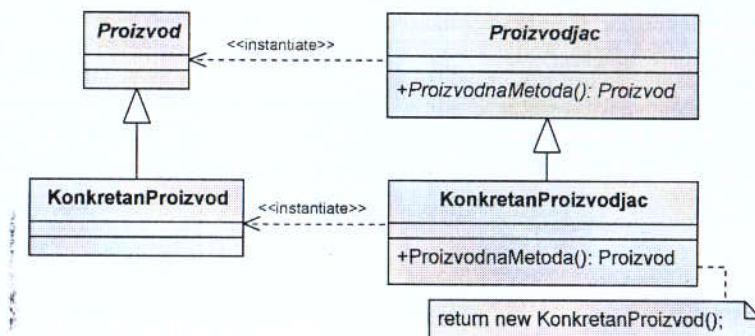
```
// @ File Name : MasSfera.java
package radionica;
import predmeti.*;
public class MasSfera extends Masina {
    private double r;
    public void MasSfera(double spTez, double r) {}
    public double r() {}
    public char vrsta() {}
    public Predmet napravi() {}
    public char vrsta() {}
    public Predmet napravi() {}
}
```

```
// @ File Name : Radnik.java
package radionica;
import predmeti.*;
public class Radnik {
    private static int posId;
    private int id = ++posId;
    private String ime;
    private int napravio;
    private Masina mas;
    public void Radnik(String ime) {}
    public int id() {}
    public String ime() {}
    public Radnik rasporedi(Masina m) {}
    public Masina dohvatiMas() {}
    public int napravio() throws GNemaMas {}
    public Predmet napravi() throws GNemaMas {}
    public String toString() {}
}
```

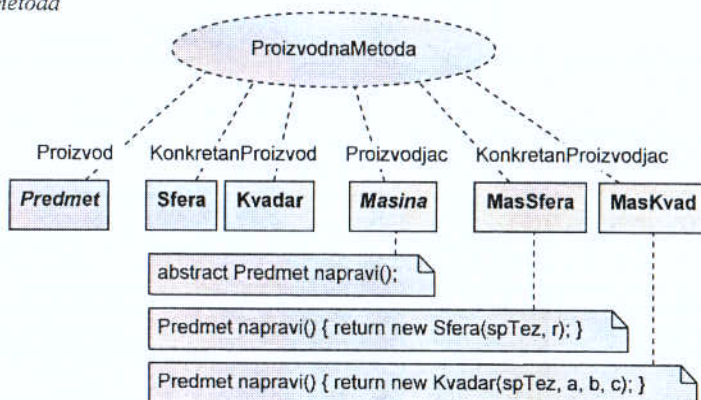
← Ne generiše se klasa za izuzetke,
ali se dodaje klauzula throws.

e) Projektni uzorak **Proizvodna metoda** (*Factory Method*)

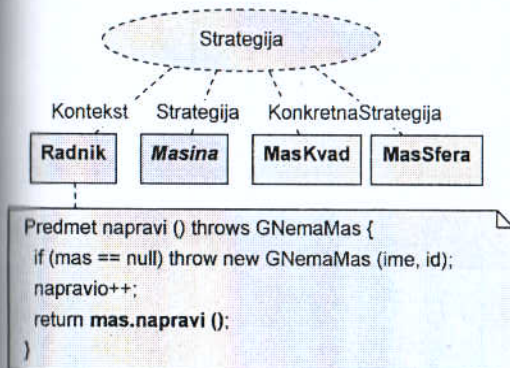
- klasni uzorak stvaranja
- predviđa interfejs za stvaranje objekata
- potklase proizvođača definišu metodu koja proizvodi konkretne objekte



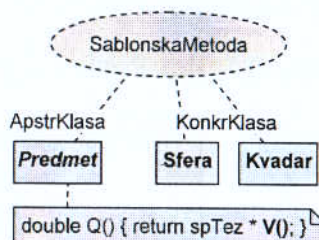
d) Primer uzorka *Proizvodna metoda*



e) Primer uzorka *Strategija*



f) Primer uzorka *Šablonska metoda*



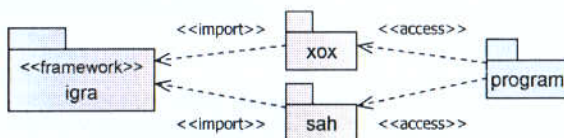
Zadatak 45 Igre na tabli (projektni uzorci *Podsetnik* i *Komanda*)

Projektovati na jeziku *UML* model sledećeg sistema:

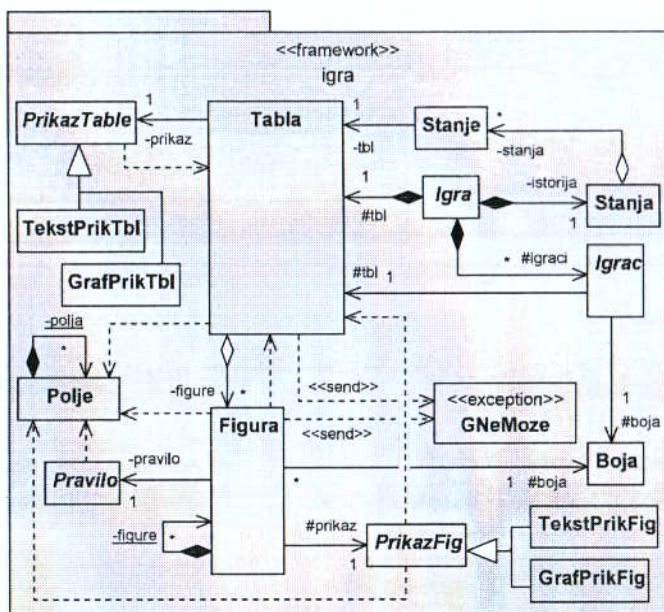
Figura za igre ima boju i može da se premešta na novo mesto na tabli za igre, uz proveravanje ispravnosti poteza u zavisnosti od igre i prirode figure. Polja dvodimenzionalne table za igre mogu biti prazna ili da sadrže jednu figuru. Figure na tabli mogu da se postavljaju, premeštaju i uklanjaju. Sadržaj table može da se prikazuje pomoću tekstualnog ili grafičkog prikazivača. Apstraktan igrač igra figurama zadate boje na zadatoj tabli. Može da odigra potez. Kod nekih igara ponekad može da dobije pravo na još jedan potez. Apstraktna igra sadrži jednu tablu za igru i nekoliko igrača. Stvara se s praznom tablom i potpunim nizom igrača. Može da se odigra jedna partija koja se sastoji od postavljanja početnog stanja igre i povlačenja poteza igrača po cikličkom redosledu do završetka igre. Partija može pre vremena da se prekine i da se napravi korak unazad proizvoljan broj puta. XoX i šah su igre koje igraju dva igrača na tabli 3×3, odnosno 8×8. Igom se upravlja programom koji koristi grafičku korisničku površ.

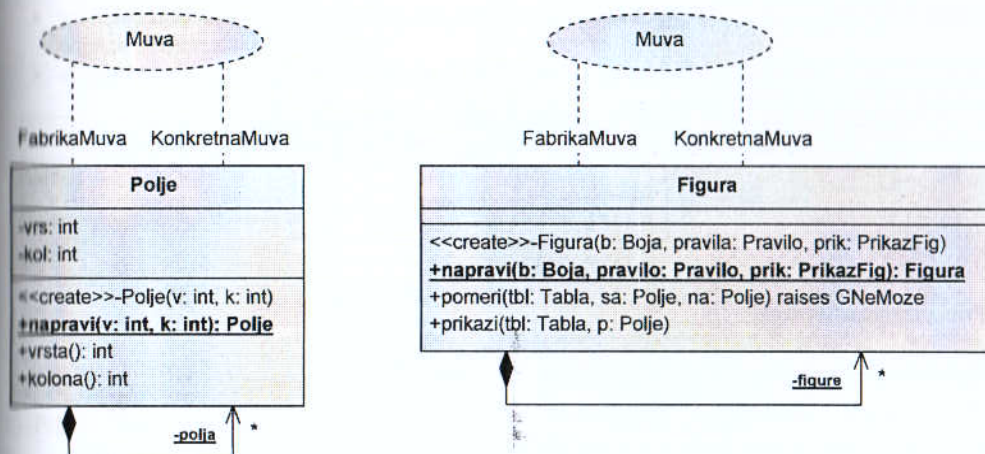
Rešenje:

a) Dijagram paketa

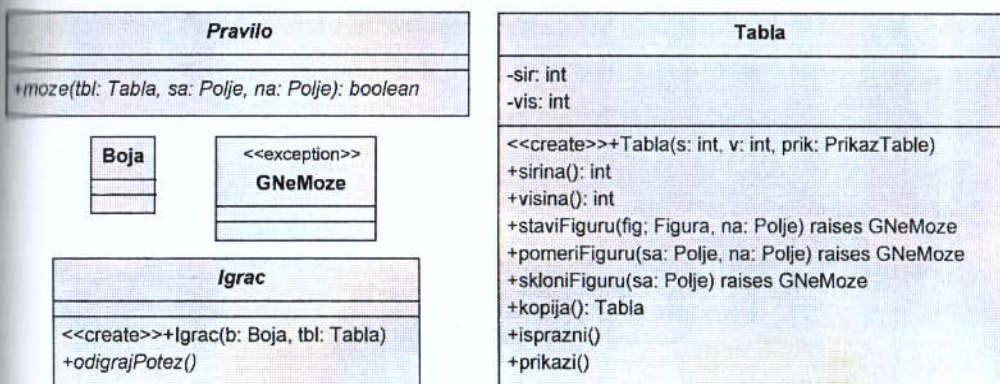


b) Dijagram klasa paketa igra

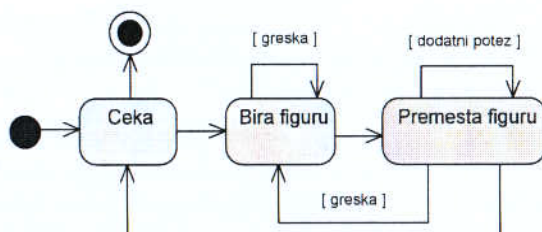


e) Primeri projektnog uzorka *Muva*

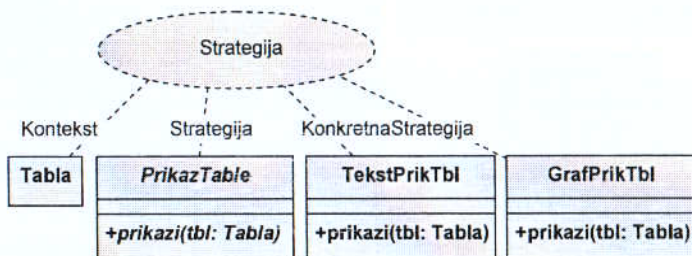
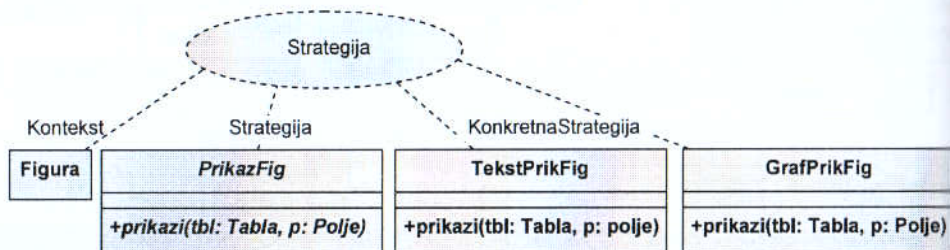
d) Ostale klase paketa igra



e) Dijagram stanja igrača

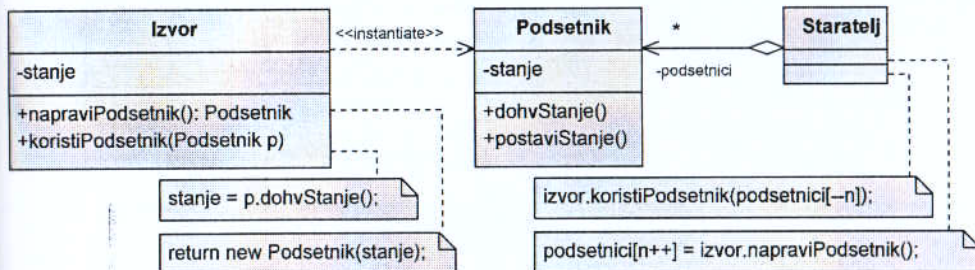
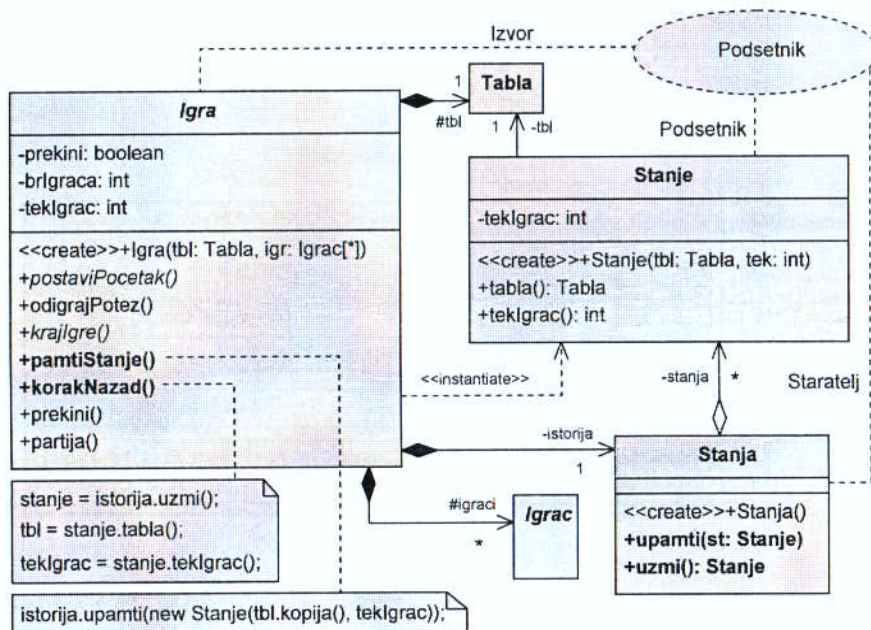


f) Primeri uzorka Strategija

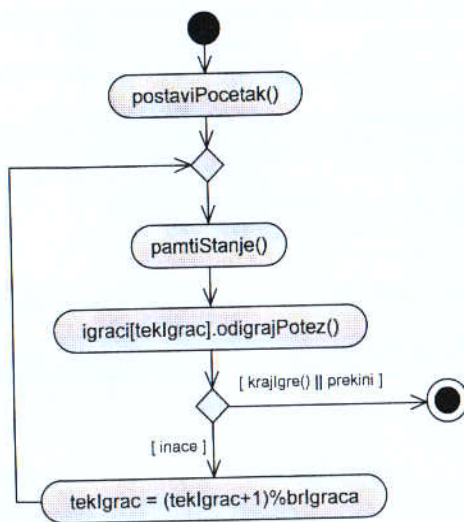


g) Projektni uzorak **podsetnik** (*memento*)

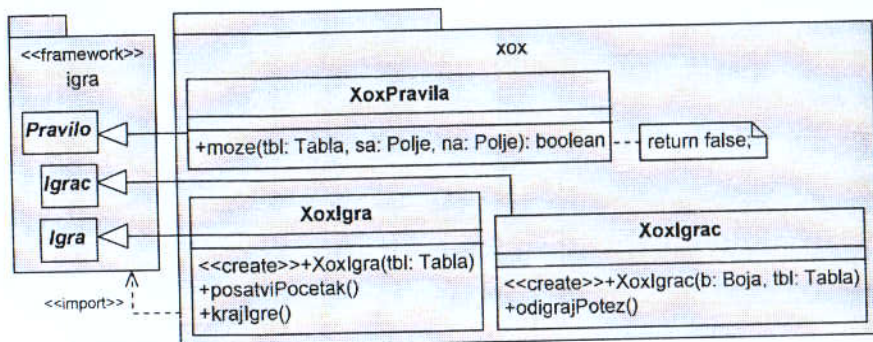
- objektni uzorak ponašanja
- snimanje unutrašnjeg stanja objekta bez narušavanja ućaurivanja podataka radi kasnijeg vraćanja objekta u ranije stanje
- staratelj traži podsetnik od izvora, čuva ga neko vreme i po potrebi vraća izvoru

h) Primer uzorka *Podsetnik*

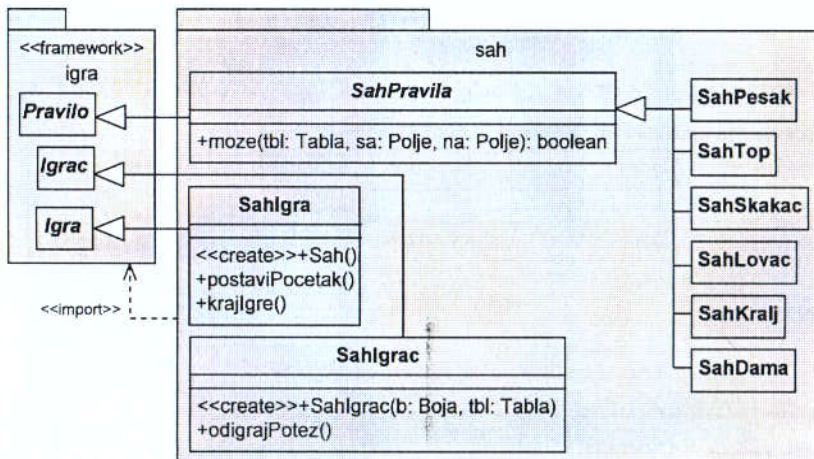
i) Dijagram aktivnosti igranja partije



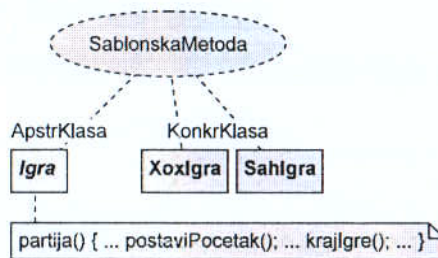
j) Paket xox



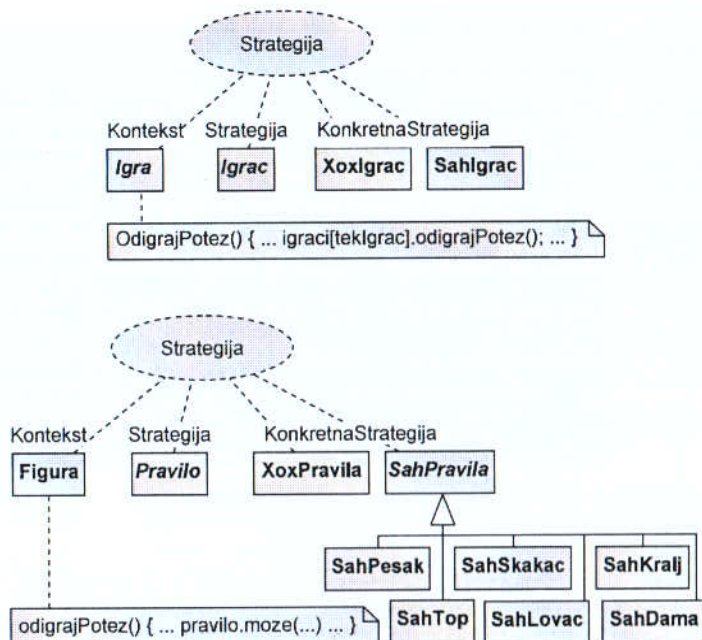
k) Paket sah



l) Primer uzorka ŠablonskaMetoda

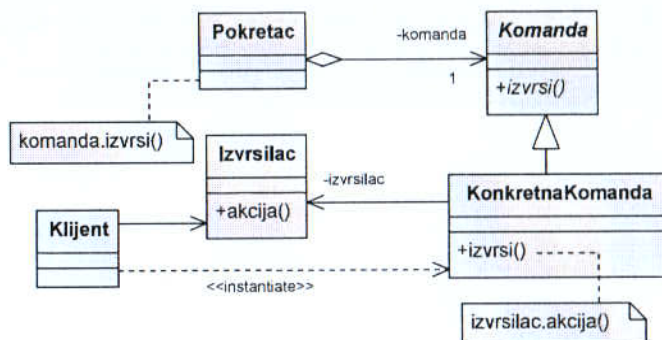


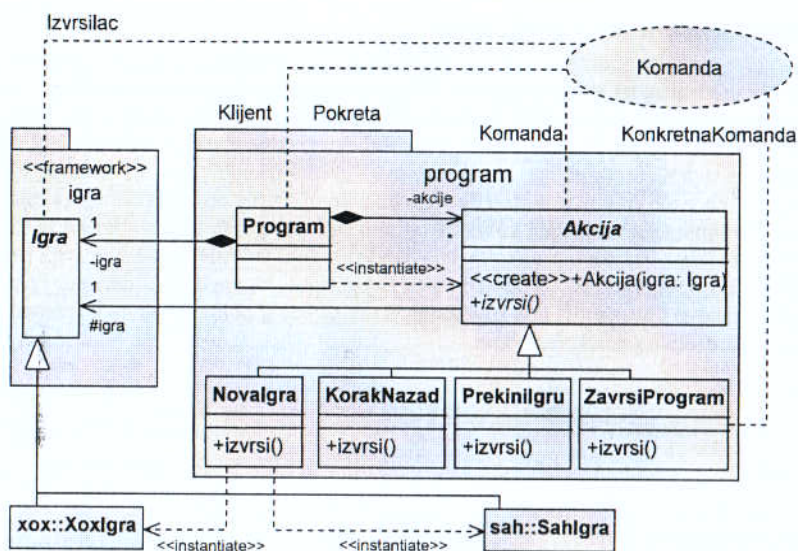
m) Još primera uzorka *Strategija*



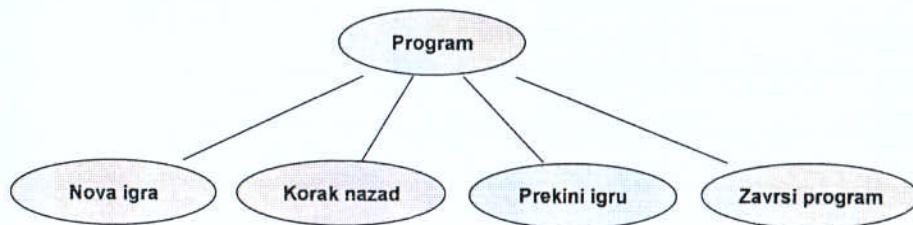
n) Projektni uzorak **Komanda** (*Command*)

- objektni uzorak ponašanja
- razdvaja izvršioca radnje od pokretača omogućujući parametrizaciju klijenta, stavljanje zahteva u redove, vođenje dnevnika i poništavanje dejstva operacija
- klijent pravi objekat komande koji prosleđuje zahtev pokretača izvršiocu



o) Primer uzorka *Komanda* u paketu *program*

p) Dijagram slučajeja korišćenja programa



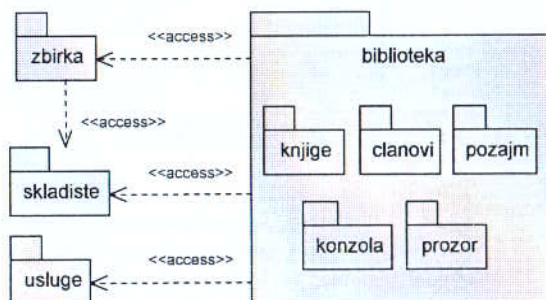
Zadatak 46 Biblioteka (dijagram komponentata, projektni uzorci *Most* i *Apstraktna fabrika*)

Projektovati na jeziku *UML* model sledećeg sistema:

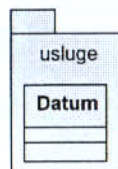
Apstraktno skladište može da sadrži apstraktne podatke koji se identifikuju pomoću apstraktnog ključa. Podaci u skladištu uređeni su po veličini ključeva. Može da se stavi neki podatak u skladište, da se ispita da li postoji podatak s datim ključem, da se podatak s datim ključem dohvati i da se izvadi. Niz ograničenog kapaciteta, lista, binarno stablo i datoteka su skladišta. Apstraktna zbirka sadrži skladište čiji sadržaj obrađuje. Biblioteka sadrži zbirku knjiga, zbirku članova i zbirku pozajmljivanja. Knjige se identifikuju na osnovu naslova. Za svaki naslov postoji jedna stavka u kojoj, pored opštih podataka o knjizi, postoji broj primeraka koje biblioteka poseduje i koliko je od njih trenutno pozajmljeno. Članovi se identifikuju pomoću šifre članske karte. Stavke o pozajmljivanju se identifikuju jedinstvenim automatski generisanim celobrojnim identifikatorom i sadrže datum uzimanja knjige i rok za vraćanje u danima. Postoje grupe obrada za knjige, članove i pozajmljivanja koje mogu da se izvode u obliku dijaloga u tekstualnom režimu ili korišćenjem prozora u grafičkom režimu.

Rešenje:

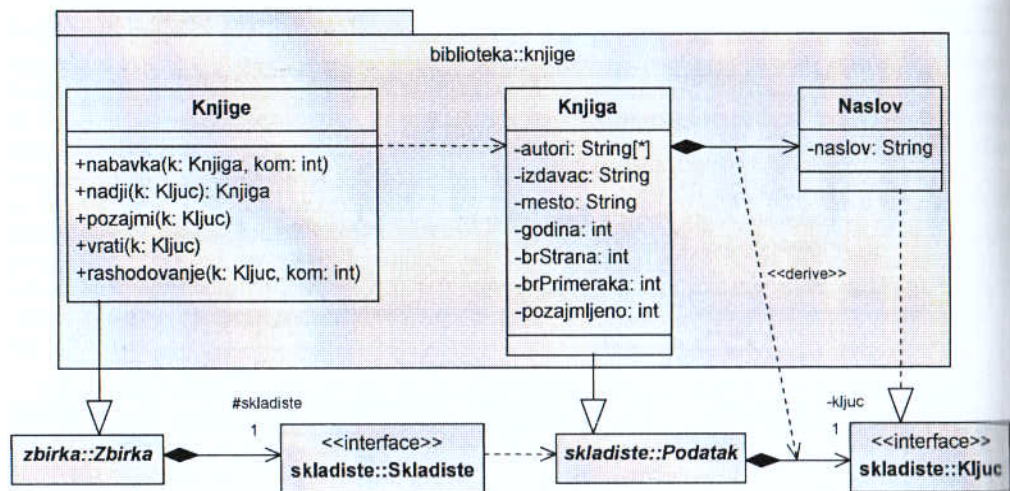
a) Dijagram paketa



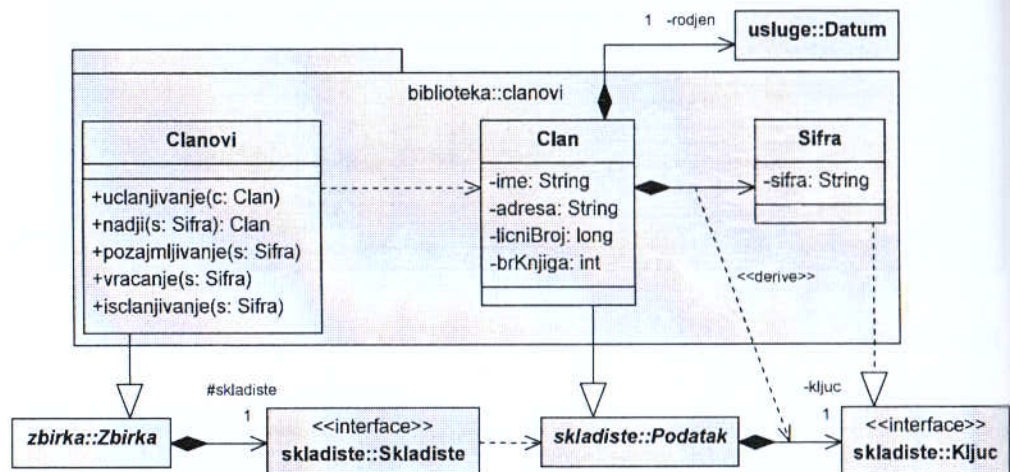
b) Paket usluge

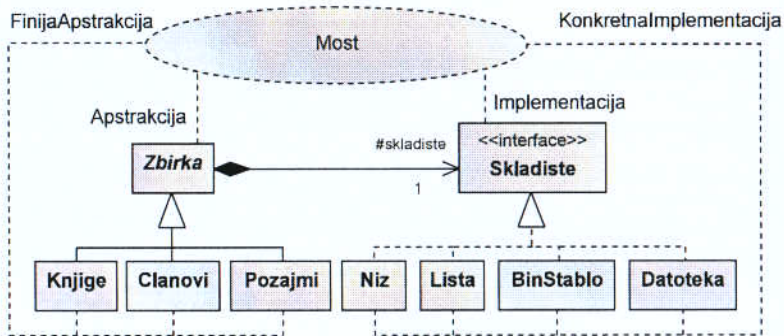


e) Paket biblioteka::knjige

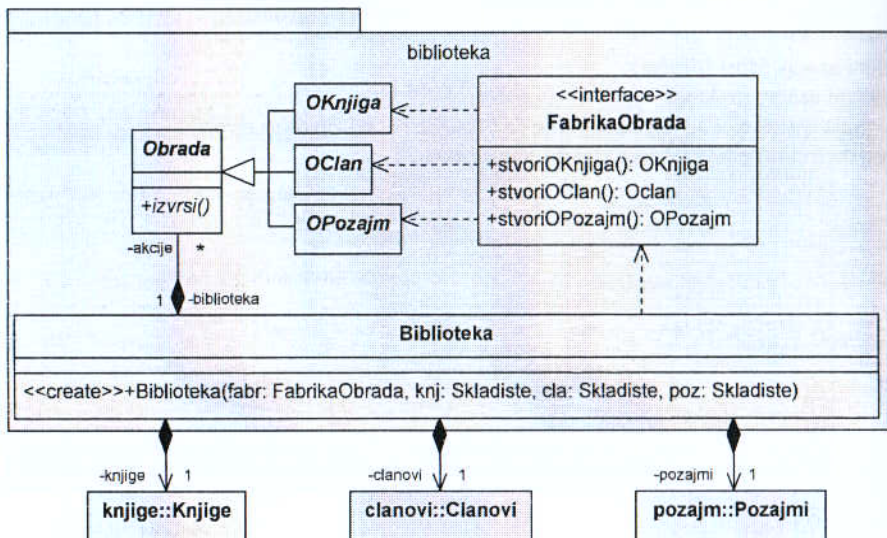


f) Paket biblioteka::clanovi



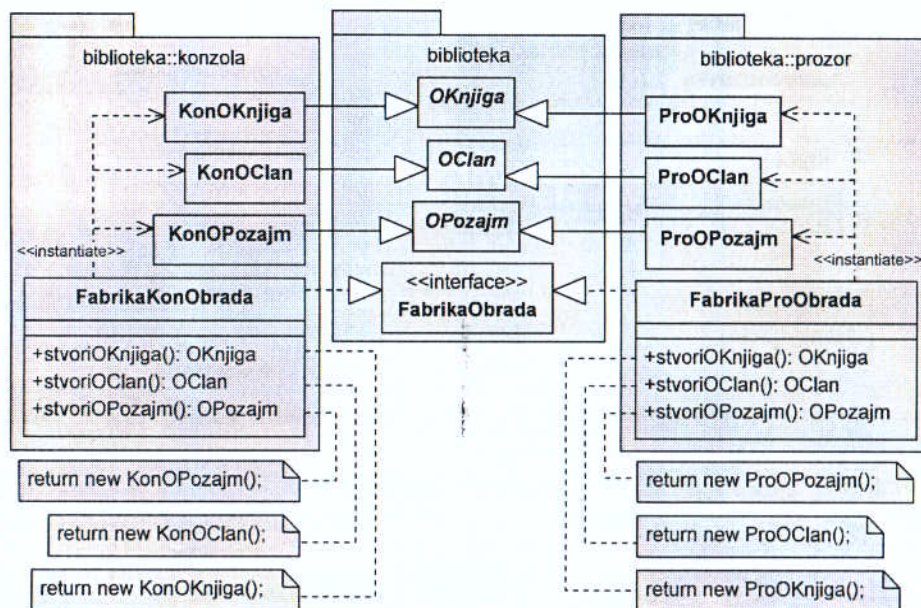
i) Primer uzorka *Most*

j) Paket biblioteka

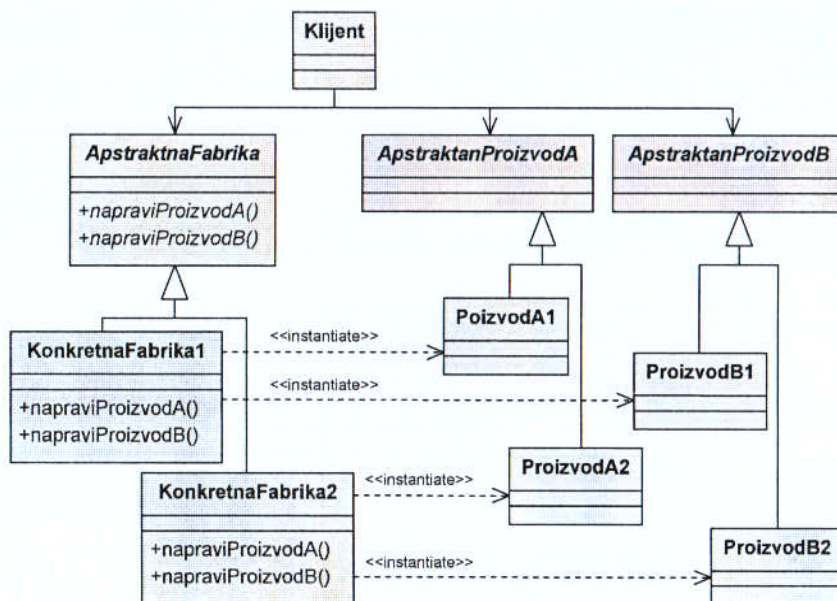


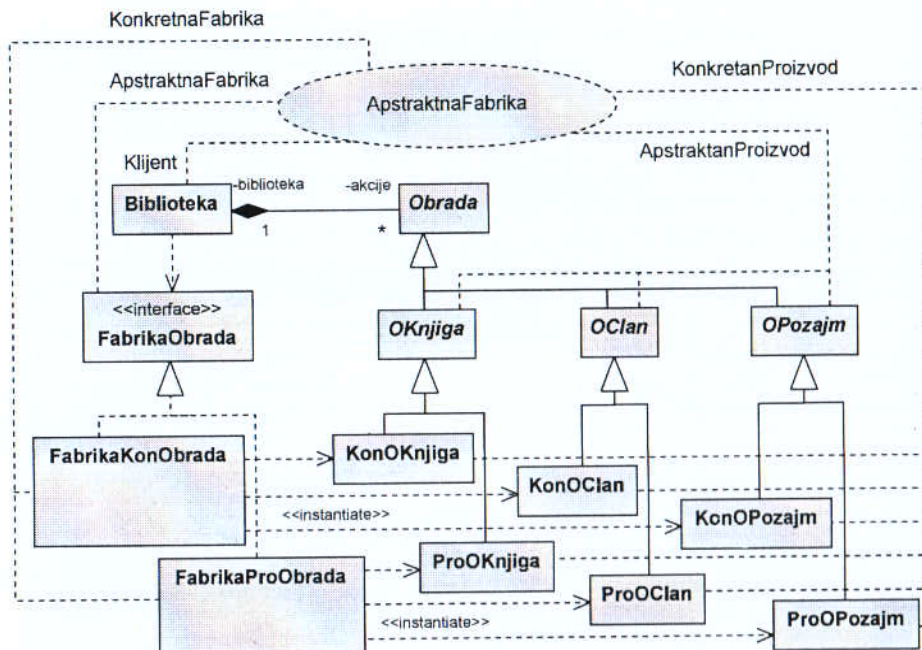
k) Paket biblioteka::konzola

l) Paket biblioteka::prozor

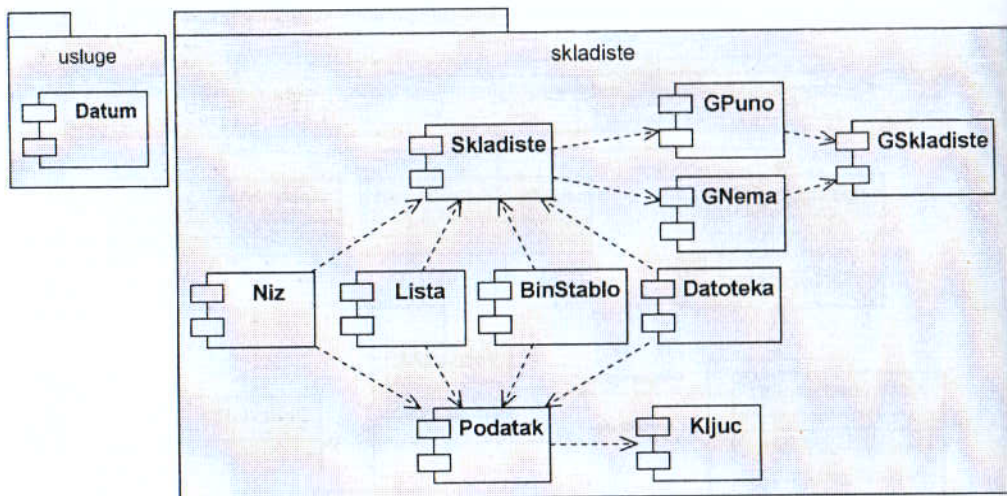
m) Projektni uzorak **Apstraktna fabrika** (*Abstract Factory*)

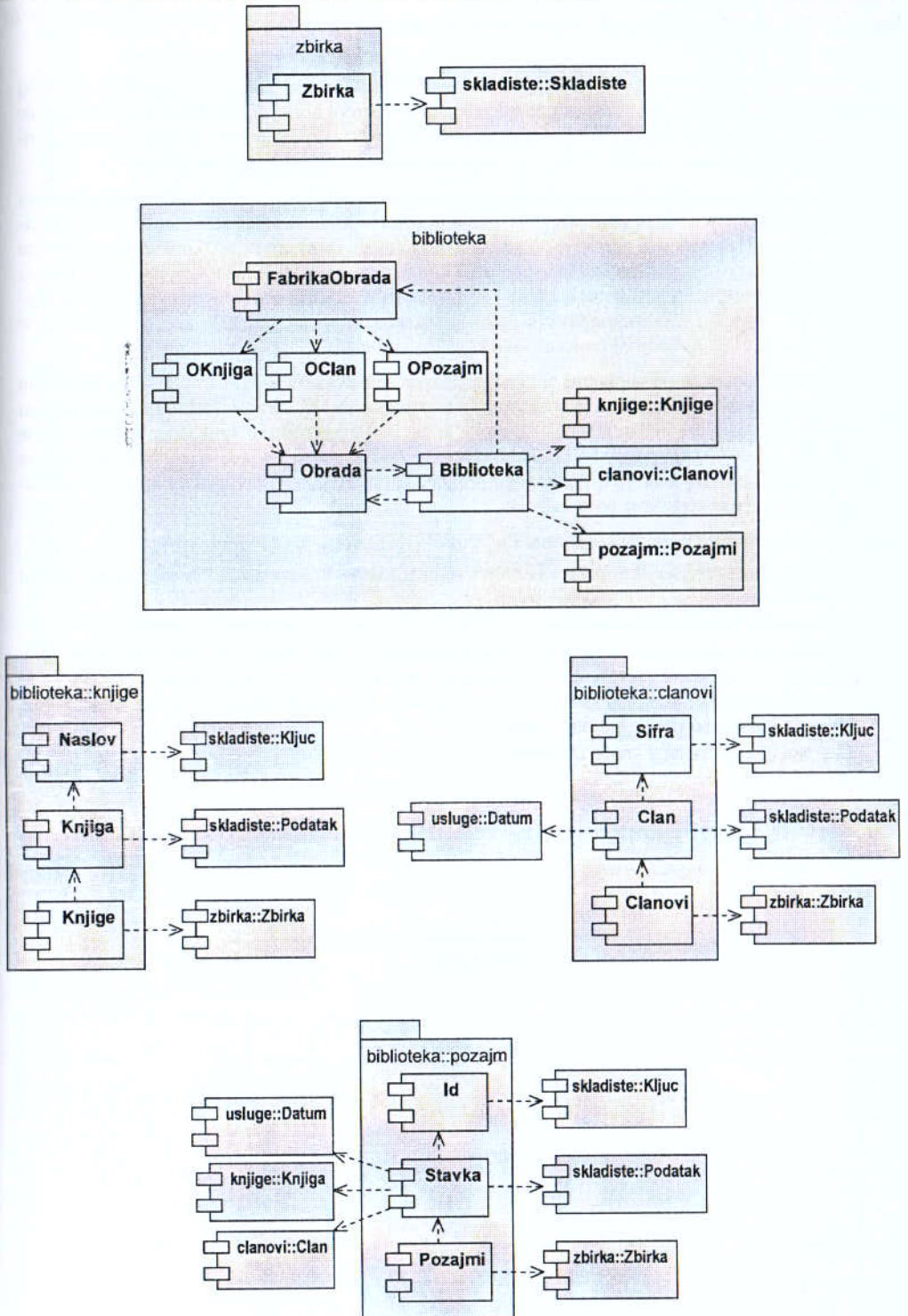
- objektni uzorak stvaranja
- omogućuje stvaranje objekata određene grupe tipova (familije vrsta proizvoda) po izboru
- klijent pravi proizvode iz određene grupe vrsta proizvoda pomoću odgovarajuće fabrike



n) Primer uzorka *Apstraktna fabrika*

o) Dijagrami komponentata





Zadatak 47 Uređaj, operacije i API (projektni uzorak *Fasada*) {I, 14.02.2007.}

Uređaj sadrži tastaturu funkcijskih dugmadi gde svako dugme ima pridruženu posebnu operaciju u svakoj aplikaciji (režimu rada uređaja). Kada se pritisne neko dugme uređaj obavesti sve registrovane rukovaoce događaja. Rukovaoci implementiraju interfejs koji predviđa dostavljanje celobrojnog identifikatora pritisnutog dugmeta.

Rukovalac na osnovu celobrojnog identifikatora dugmeta i tabele preslikavanja određuje operaciju koju treba izvršiti i smešta je u red čekanja. Operacije koje je smestio jedan rukovalac izvršavaće se sekvencijalno, po redosledu smeštanja, a operacije raznih rukovalaca mogu da se izvršavaju konkurentno. Svakom redu operacija pridružen je aktivan izvršilac operacija koji uzima operacije iz reda i izvršava ih ne znajući ništa o prirodi tih operacija, niti o načinu njihovog izvršavanja. Složene operacije nisu zavisne od konkretnih aplikacija i to su sekvenca, grananje i petlja. Petlja sadrži jednu, grananje dve, a sekvenca više operacija. Grananje i petlja sadrže i po jedan logički uslov, koji može biti proizvoljne složenosti.

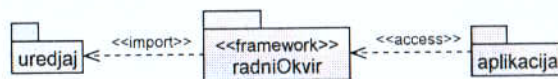
Proste operacije zavisne su od konkretne aplikacije, tako da se mora obezbediti da se za jednu aplikaciju kreira konzistentan skup operacija od interesa, koji se u vreme izvršavanja može zameniti drugim skupom operacija. Konkretni uslov nekog grananja ili petlje je zavisan od konkretne aplikacije, a može i da se promeni u vreme izvršavanja programa. Svaka operacija se izvršava tako što poziva jednu ili više metoda aplikativnog programskog interfejsa. Aplikativni programski interfejs predstavlja skup operacija aplikativne logike koja se ostvaruje kroz proizvoljan broj klasa.

Projektovati na jeziku UML prethodni sistem. Priložiti:

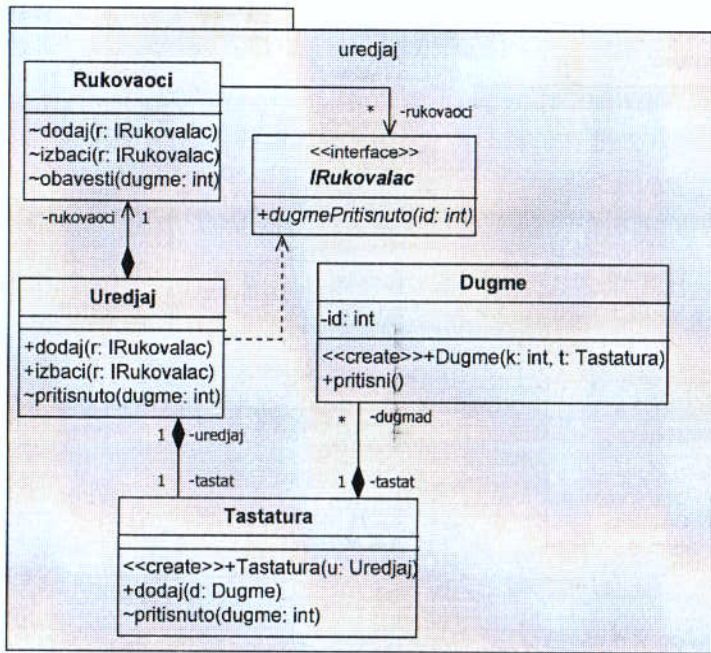
- dijagram klasa razvrstanih u pakete (detaljan sadržaj klasa i odnose među klasama na potrebnom broju slika),
- prikaz korišćenih projektnih uzoraka,
- dijagram objekata koji prikazuje uređaj sa dva dugmeta, red koji sadrži jednu sekvencu s nekoliko operacija i izvršioca koji izvršava petlju koja sadrži grananje s dve proste operacije,
- dijagram sekvence i dijagram aktivnosti za ceo scenario koji usledi nakon pritiska na jedno dugme kojim se zahteva izvršavanje proste operacije,
- dijagram komponenata u vreme izvršenja.

Rešenje:

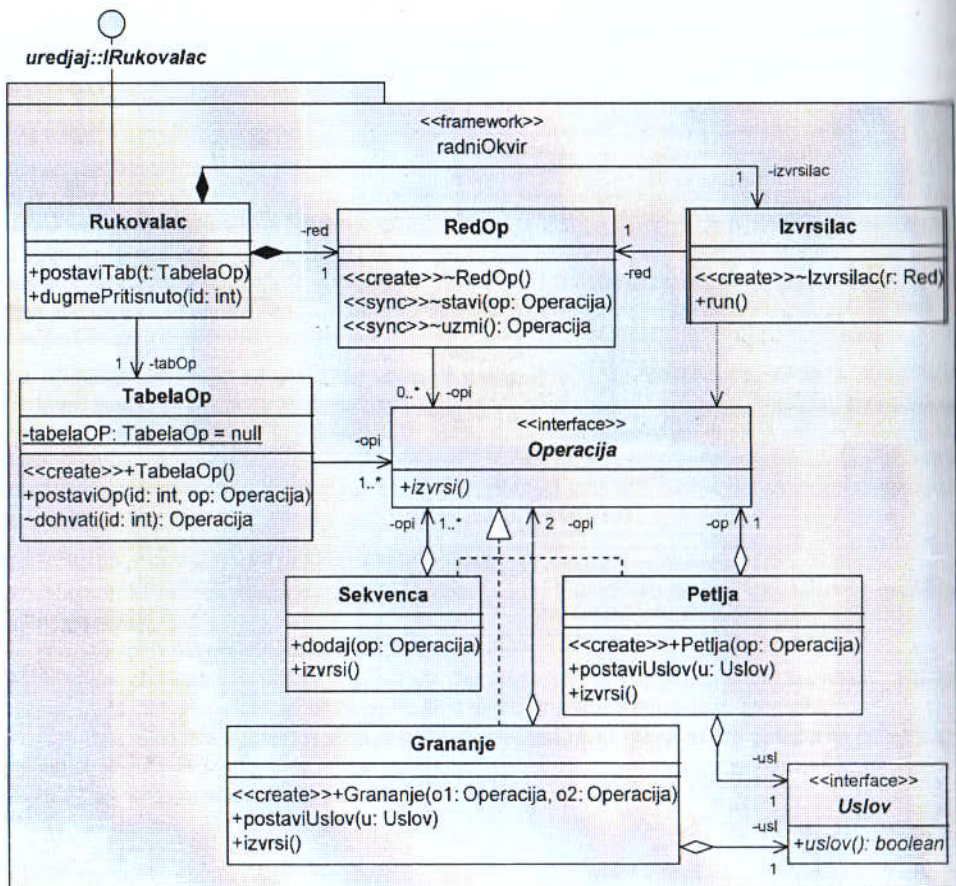
a) Dijagram paketa



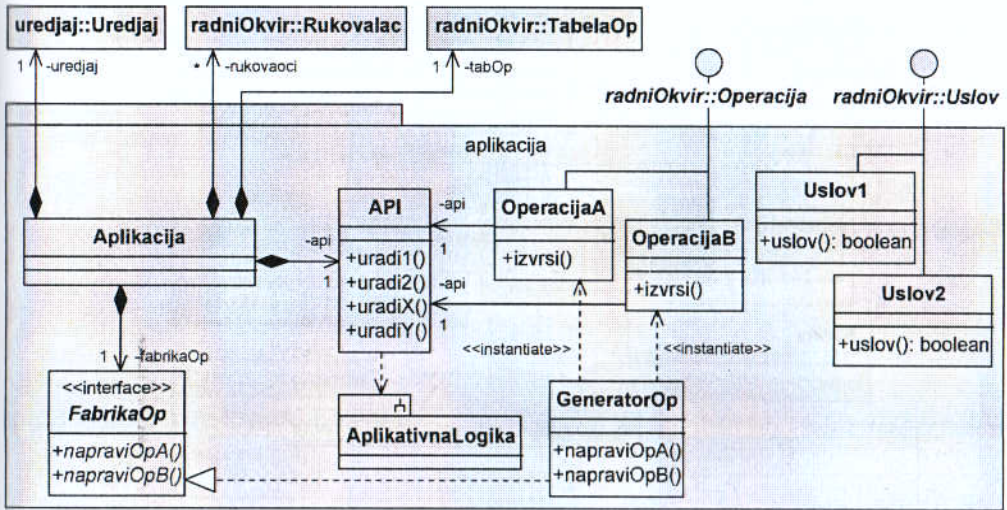
b) Dijagram klasa paketa uredjaj



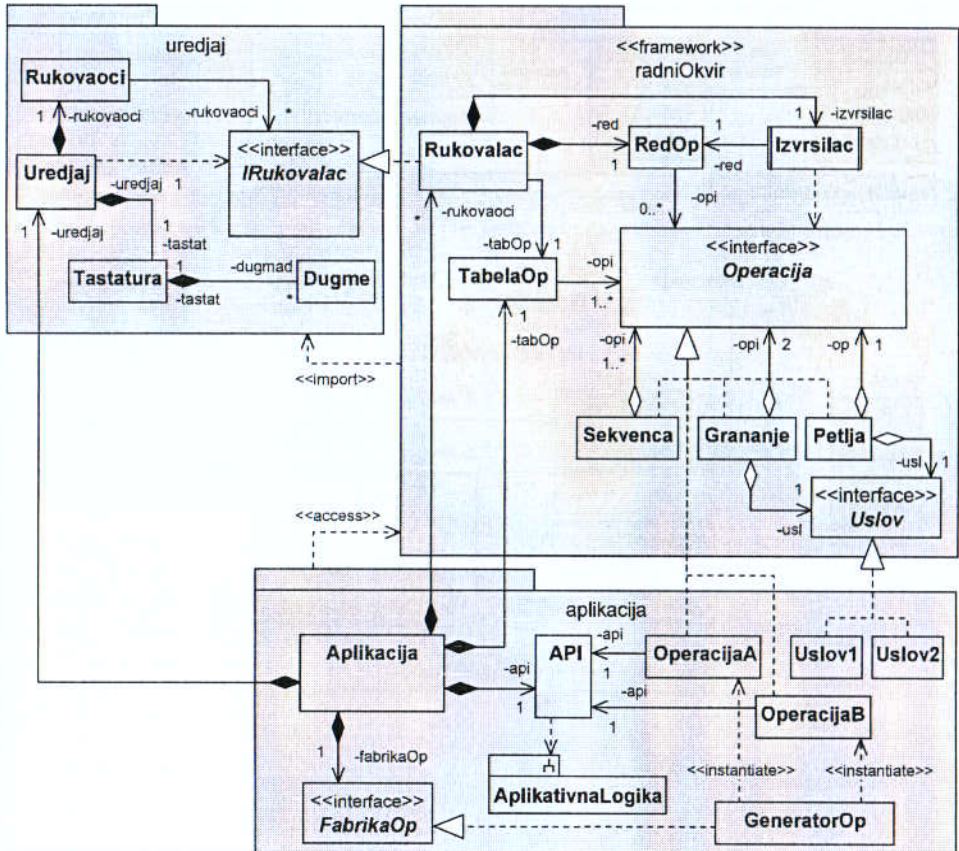
c) Dijagram klasa paketa radniOkvir



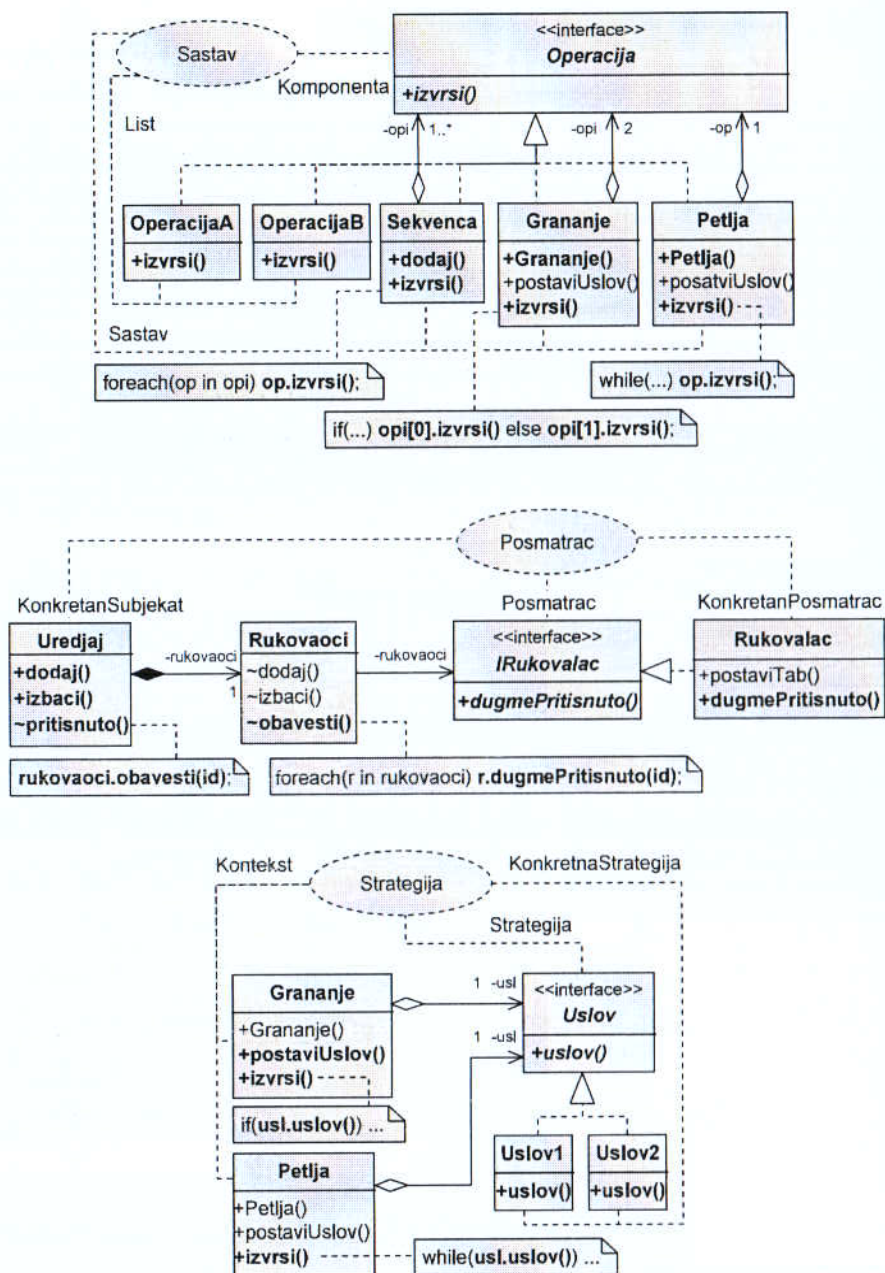
d) Dijagram klasa paketa aplikacija

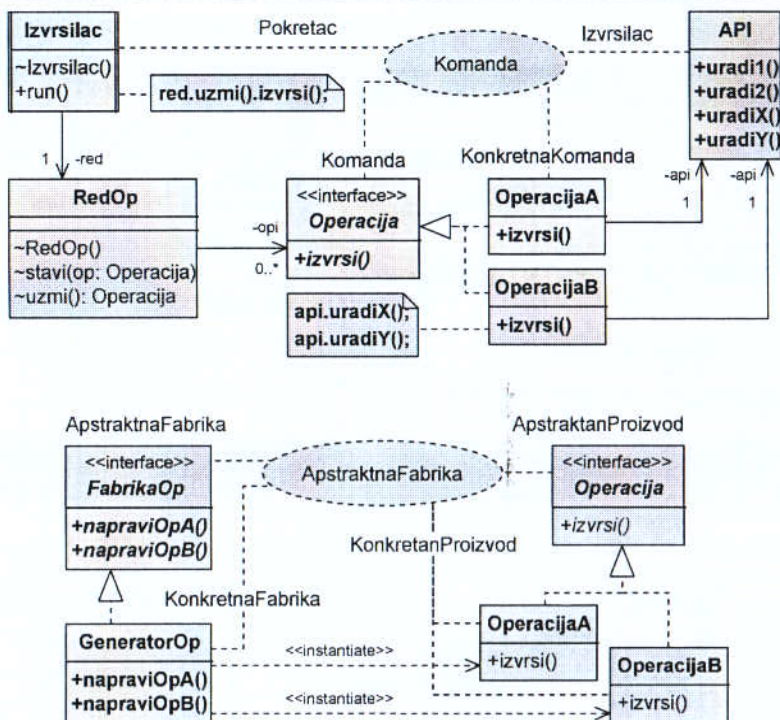


e) Pregledni dijagram klasa



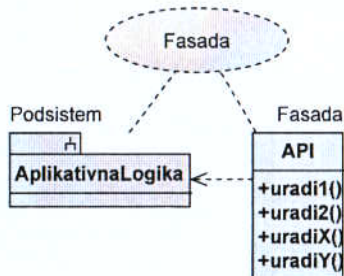
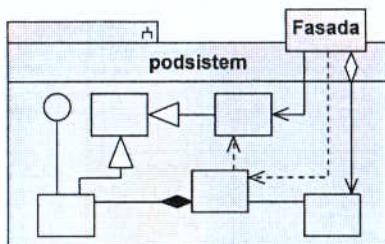
f) Projektni uzorci



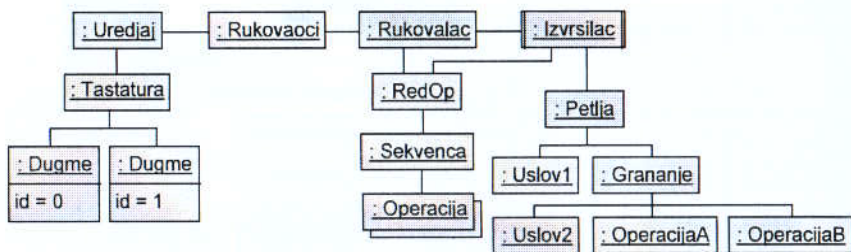


g) Projektni uzorak **Fasada** (*Facade*)

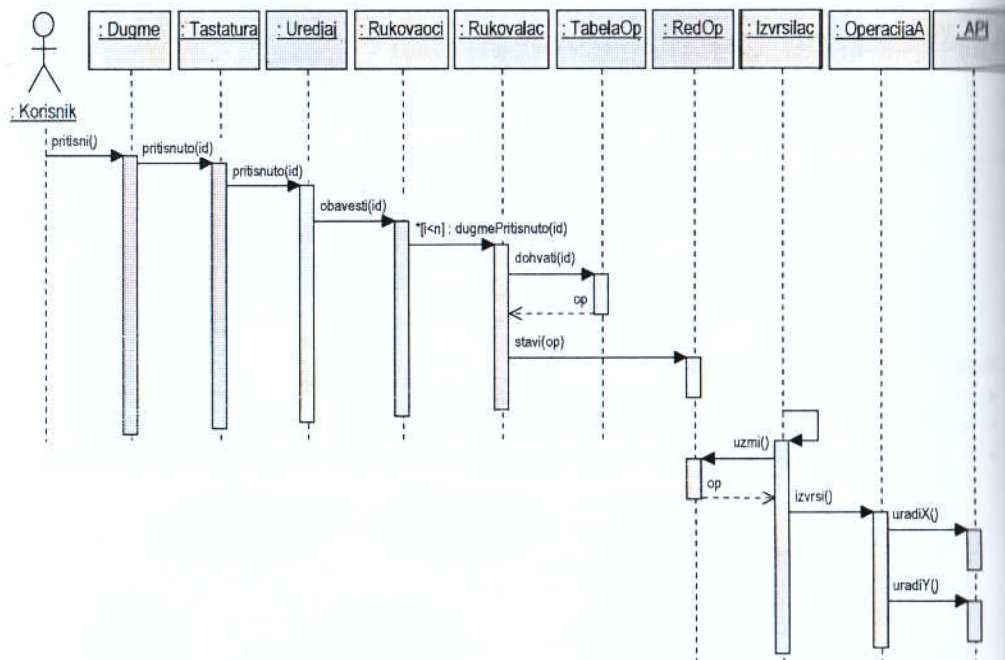
- objektni uzorak strukture
- daje interfejs višeg nivoa za skup interfejsa jednog podsistema
- klijent podsistem koristi preko fasade, ali nije sprečen ni da pristupa klasama podsistema direktno



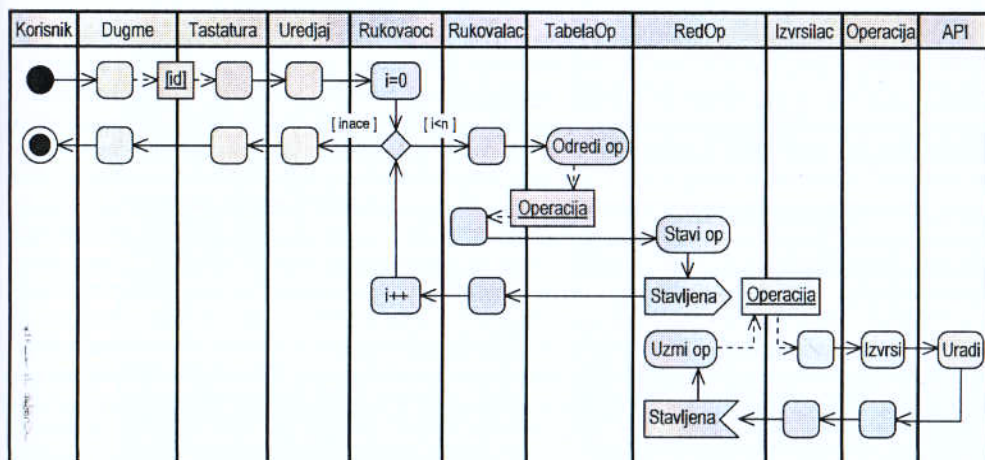
h) Dijagram objekata



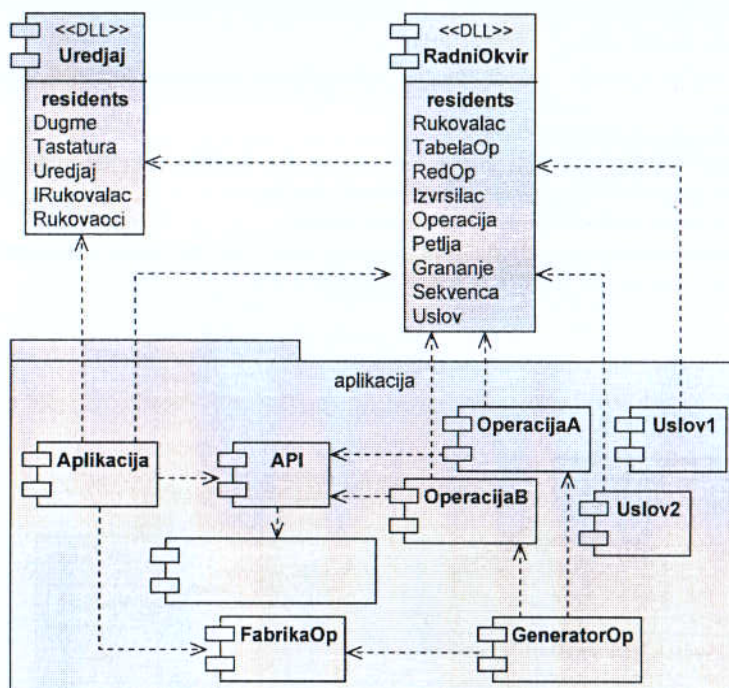
i) Dijagram sekvence prilikom pritiska na dugme



j) Dijagram aktivnosti prilikom pritiska na dugme



k) Dijagram komponenta u vreme izvršenja



Zadatak 48 Datoteke, sistem datoteka i medijumi (projektni uzorak **Zastupnik**) fl, 20.01.2010.)

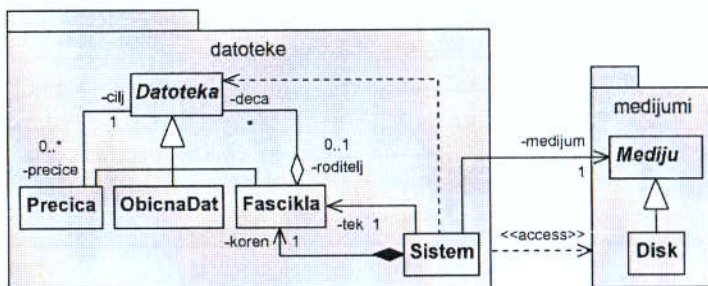
Datoteka ima ime, veličinu, vreme stvaranja i vreme poslednjeg menjanja koji mogu da se dohvate. Vreme se predstavlja dugačkim celim brojem. Datoteka može da se otvori za pristup i uništi. Obična datoteka je datoteka u koju može da se upisuje i iz koje može da se čita sadržaj. Fascikla je datoteka koja sadrži proizvoljan broj datoteka. Prečica je datoteka koja samo ukazuje na zadatu ciljnu datoteku i sadrži puno ime ciljne datoteke. Više prečica mogu da ukazuju na istu datoteku. Sistem datoteka sadrži jednu fasciklu koja predstavlja koren celokupnog sistema datoteka. Jedna od fascikli u sistemu je tekuća fascikla i sve operacije u sistemu se podrazumevano izvode u toj fascikli. Svaka datoteka pripada nekoj fascikli koja se zadaje prilikom stvaranja datoteke i naziva se roditeljem datoteke. Može da se odredi puno ime datoteke koje sadrži imena fascikli od korene fascikle sistema do neposrednog roditelja datoteke i imena same datoteke. Datoteka može da se preimenuje, da se kopira ili premesti u drugu fasciklu. Prilikom premeštanja, odnosno uništavanja datoteke, moraju se ažurirati, odnosno uništiti sve prečice koje ukazuju na tu datoteku. Fascikli može, jedna po jedna, da se doda i izbaci proizvoljan broj datoteka, da se pronade datoteka zadatog imena i da se odredi ukupna veličina svih sadržanih datoteka. Otvaranjem fascikle ista postaje tekuća fascikla sistema. Prilikom stvaranja prečice zadaje se i ciljna datoteka na koju ona pokazuje. Otvaranje prečice podrazumeva otvaranje ciljne datoteke. Sistem datoteka ima ime. Može da se dohvati ime i kapacitet sistema i da se odredi ukupna veličina svih datoteka u sistemu. Sistemu, u tekućoj fascikli, može da se doda nova datoteka, da se otvori ili uništi datoteka zadatog imena i da se odredi ukupna veličina tekuće fascikle. Sistem se uskladištava na medijumu koji ima određeni kapacitet koji može da se dohvati. Na medijumu može da se zauzme i oslobodi prostor zadate veličine, da se dohvati ukupna veličina trenutno zauzetog prostora na medijumu, da se čita i piše po medijumu. Disk je medijum.

Projektovati na jeziku UML prethodni sistem. Priložiti:

- dijagram klasa razvrstanih u pakete (detaljan sadržaj klasa i odnose među klasama na potrebnom broju slika),
- prikaz korišćenih projektnih uzoraka,
- dijagram objekata koji prikazuje sistem datoteka sa dve fascikle, nekoliko običnih datoteka i klijenta koji upravo želi da pristupa datoteci pomoću prečice,
- dijagram aktivnosti premeštanja datoteke u drugu fasciklu,
- dijagram komponenata pod pretpostavkom da svakoj klasi odgovara posebna komponenta (na primer, `.class` datoteka prevedenog programa na jeziku *Java*).

Rešenje:

a) Dijagram klasa



datoteke::Datoteka

-ime: String
 -vel: long
 -stvorena: long
 -menjana: long

<<create>>+Datoteka(ime: String, f: Fascikla)
 +ime(): String
 +velicina(): long
 +stvorena(): long
 +menjana(): long
 +otvori()
 +unisti()
 +punolme(): String
 +dodaj(p: Precica)
 +izbaci(p: Precica)
 +preimenuj(novolme: String)
 +kopiraj(odrediste: Fascikla)
 +premesti(odrediste: Fascikla)

datoteke::Precica

-punolmeCilja: String

<<create>>+Precica(ime: String, f: Fascikla, cilj: Datoteka)
 +unisti()
 +otvori()
 +azuriraj(novoPunolmeCilja: String)

datoteke::ObicnaDat

<<create>>+ObicnaDat(ime: String, f: Fascikla)
 +otvori()
 +unisti()
 +pisi(niz: byte[], duz: long)
 +citaj(kolicina: long): byte[]

datoteke::Fascikla

<<create>>+Fascikla(ime: String, f: Fascikla)
 +otvori()
 +unisti()
 +kopiraj(odrediste: Fascikla)
 +dodaj(dat: Datoteka)
 +izbaci(dat: Datoteka)
 +nadji(ime: String): Datoteka
 +ukVel(): long

datoteke::Sistem

-ime: String
 -kap: long

<<create>>+Sistem(ime: String, m: Medijum)
 +ime(): String
 +kapacitet(): long
 +zauzeto(): long
 +dodaj(d: Datoteka)
 +otvori(ime: String): Datoteka
 +unisti(ime: String)
 +ukVel(): long

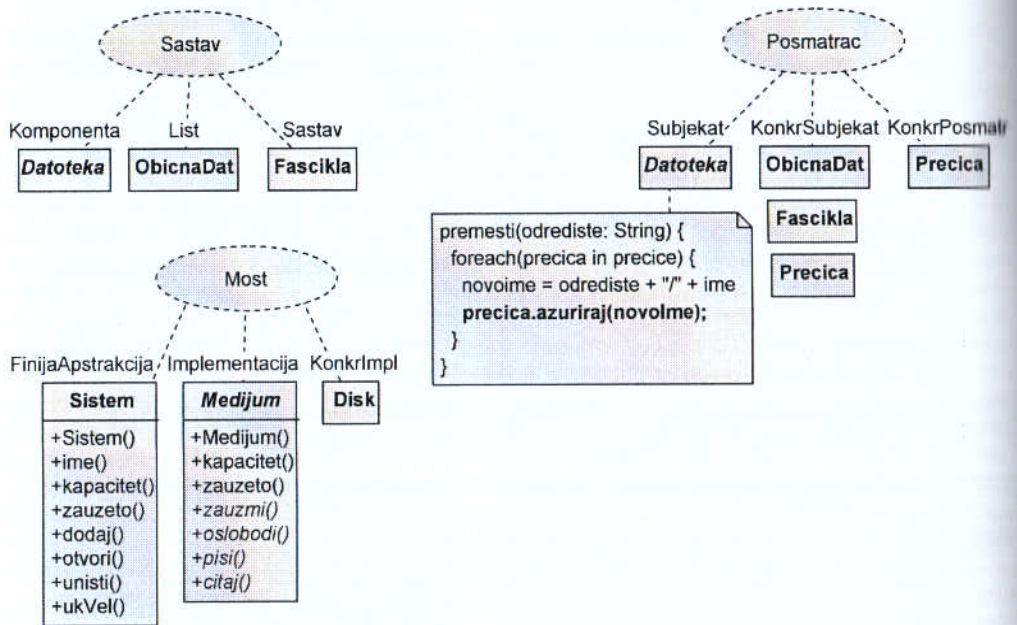
medijumi::Medijum

-kap: long
 -zauzeto: long

<<create>>+Medijum(kap: long)
 +kapacitet(): long
 +zauzeto(): long
 +zauzmi(vel: long)
 +oslobodi(vel: long)
 +pisi(niz: byte[], duz: long)
 +citaj(kolicina: long): byte[]

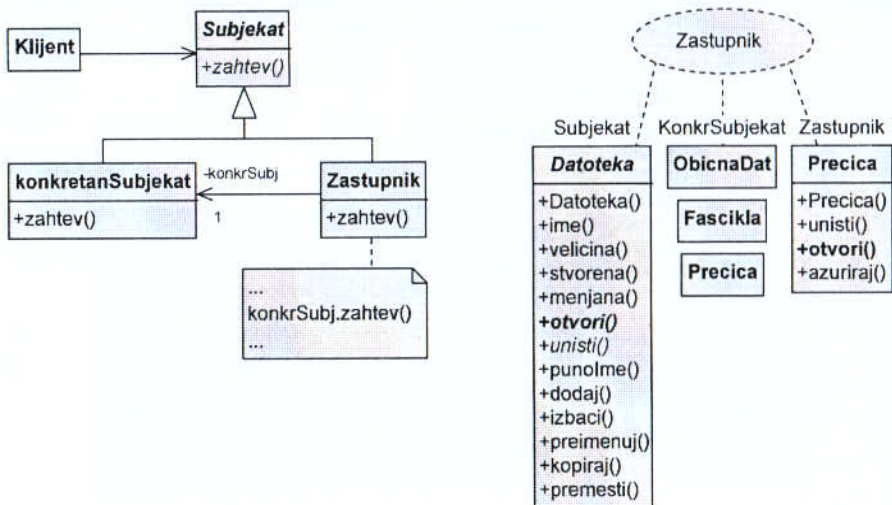
medijumi::Disk

b) Projektni uzorci



c) Projektni uzorak Zastupnik (Proxy)

- objektni uzorak strukture
- predstavlja zamenu nekog objekta, često radi ostvarivanja kontrole pristupa tom objektu
- klijent pristupa pravom objektu preko objekta zastupnika



Zadatak 49 Osoba, zaposleni, firma, roman, autor, izdavačka kuća, kurir, izvršioci i nalog (projektni uzorci *Posrednik i Lanac odgovornosti*) [1, 10.01.2011.]

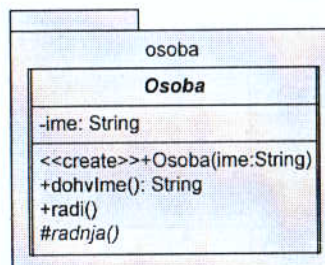
Aktivna osoba ima ime koje može da se dohvatiti. Ciklički izvršava neku radnju. Zaposleni je osoba koja radi u zadatoj firmi. Datum zapošljavanja predstavlja se u obliku celog broja. Može da se dohvatiti naziv njegovog zaduženja i datum zapošljavanja. Firma ima zadat naziv. Može da zaposli i otpusti po jednog zaposlenog. Zapošljavanje može da ne uspe. Svi zaposleni u firmi mogu da se obilaze po redosledu zapošljavanja i po redosledu imena. Roman ima zadatog autora, naslov i sadržaj koji mogu da se dohvate. Autor je osoba koja može da napiše roman koji predaje nekoj ranije izabranoj izdavačkoj kući. Izdavačka kuća je firma koja zapošljava kurire i izvršioce. Urednik, recenzent, lektor, korektor i slagač su izvršioци koji obrađuju roman. Izdavačka kuća može da primi neki roman za koji napravi nalog za obradu kojeg stavlja u zbirku naloga. Nalog sadrži podatak o romanu, statusu obrade romana i podatak o izvršiocu tekuće obrade. Mogući statusi, kroz koje se prolazi po navedenom redosledu, su: nov, recenzija, lektura, korektura, slaganje, gotov ili odbijen. Status i izvršilac mogu da se postave i dohvate. Urednik uzima po jedan nalog iz zbirke naloga i predaje ga kuriru. Kurir, na osnovu trenutnog statusa obrade, dostavlja nalog prvom izvršiocu odgovarajuće vrste. Ako je izvršilac zauzet on predaje nalog sledećem izvršiocu iste vrste. Ako je i poslednji izvršilac te vrste zauzet on vraća nalog kuriru koji ga dostavlja uredniku. Urednik ga vraća u zbirku naloga. Ako postoji slobodan izvršilac isti izvrši svoju obradu i nalog predaje kuriru. Ako je obrada uspešno završena kurir dostavlja nalog prvom izvršiocu sledeće obrade, inače dostavlja uredniku. Ako se sve obrade završe uspešno po dostavljanju naloga uredniku, on stavlja roman u zbirku romana izdavačke kuće. Po završetku obrade romana (bilo uspešno ili neuspešno), urednik vraća obrađeni roman autoru.

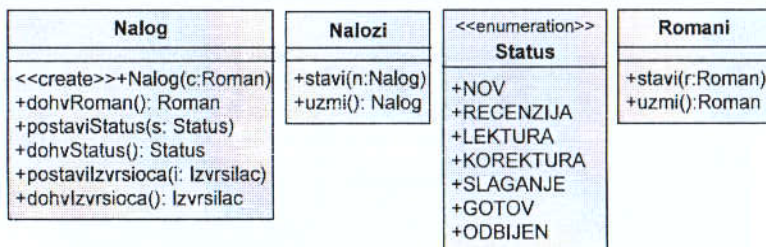
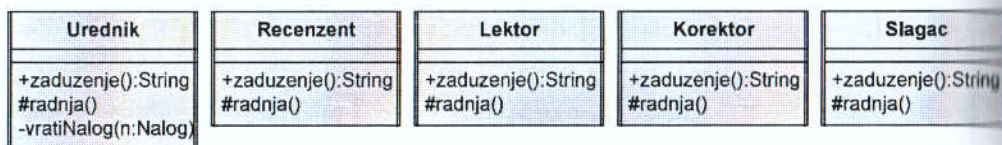
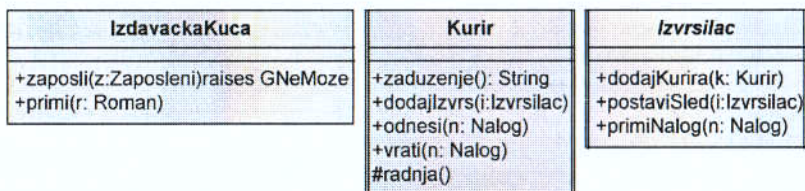
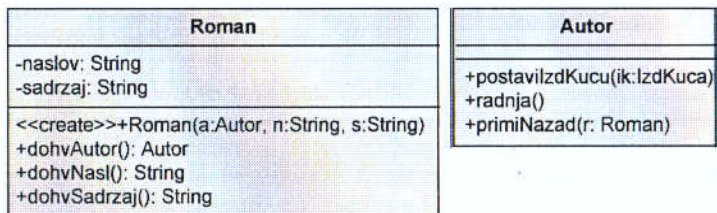
Projektovati na jeziku UML prethodni sistem. Priložiti:

- dijagram klasa razvrstanih u pakete (detaljan sadržaj klasa i odnose među klasama na potrebnom broju slika);
- prikaz korišćenih projektnih uzoraka;
- dijagram aktivnosti (s plivačkim stazama) stvaranja i prijema jednog romana od strane izdavačke kuće (samo do smeštanja u zbirku naloga);
- dijagram komponenta stavljajući srodne klase u istu komponentu.

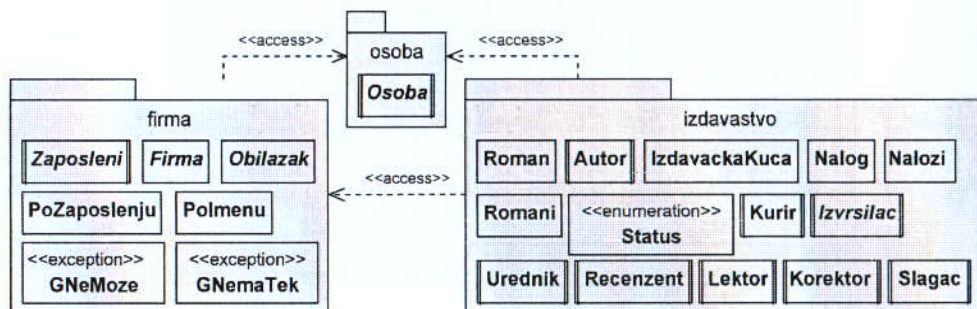
Rešenje:

a) Dijagrami klasa

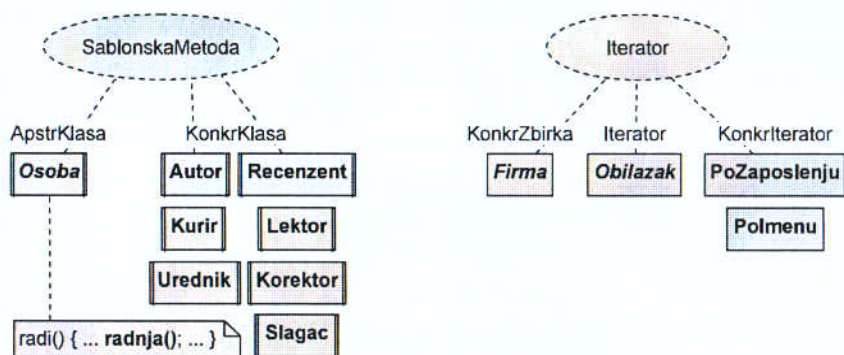




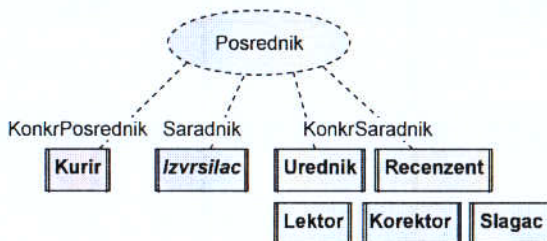
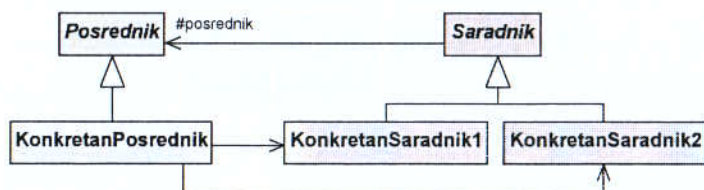
b) Dijagram paketa



e) Projektni uzorci

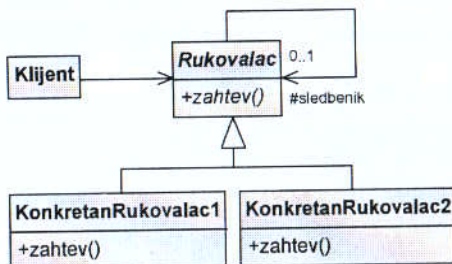
d) Projektni uzorak **Posrednik** (*Mediator*)

- objektni uzorak ponašanja
- potpomaže slabo povezivanje, jer sprečava međusobno vezivanje objekata i omogućava nezavisno menjanje njihovih međudejstava
- saradnici šalju i primaju zahteve, odnosno rezultate zahteva, preko posrednika

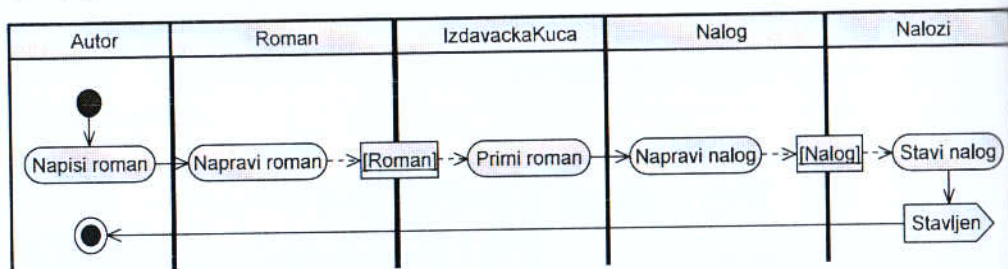


e) Projektni uzorak **Lanac odgovornosti** (*Chain of responsibility*)

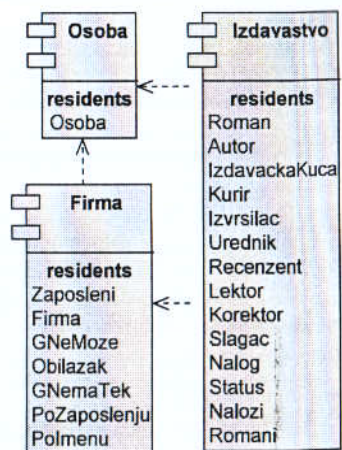
- objektni uzorak ponašanja
- omogućuje razdvajanje pošiljaoca i primaoca zahteva dajući šansu da više objekata rukuje obradom zahteva
- klijent inicira obradu zahteva koji putuje duž lanca rukovalaca dok ne bude obrađen



f) Dijagram aktivnosti stvaranja i prijema romana



g) Dijagram komponenata



Zadatak 50 Preduzeće, radnici, skladište i automobili (projektni uzorak **Graditelj**) {1, 06.02.2009.}

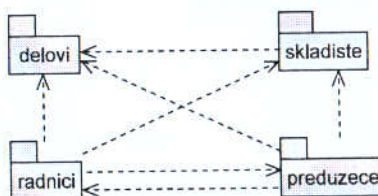
Proizvodu može da se dohvati vrsta. Automobil, deo i element su proizvodi. Automobil se sastoji od delova, a delovi od elemenata. Automobil ima serijski broj. Karoserija, točak i motor su delovi. Šraf, šipka i ploča su elementi. Skladište sadrži proizvoljan broj proizvoda. Može da se dodaje jedan proizvod i da se proizvod zadate vrste izvadi iz skladišta. Sadržaj skladišta može da se pretražuje na više načina da bi se pronašla željena vrsta proizvoda. Aktivan radnik ima ime, može da mu se dohvati kvalifikacija, može da se zaposli u nekom preduzeću i može da proizvodi proizvode koje stavlja u zadato skladište. Postoje nekvalifikovani, kvalifikovani i visokokvalifikovani radnici. Nekvalifikovani radnik proizvodi elemente na osnovu zadatog originala. Kvalifikovani radnik proizvodi delove od elemenata iz skladišta. Specijalizovani kvalifikovani radnici proizvode karoserije, točkove ili motore. Delovi se različito proizvode za različite marke automobila. Visokokvalifikovani radnik proizvodi automobile tako što im ugradi karoseriju, četiri točka i motor iz skladišta. Preduzeće zapošljava radnike, poseduje skladište i proizvodi automobile marke koja se određuje prilikom stvaranja preduzeća.

Projektovati na jeziku UML prethodni sistem. Priložiti:

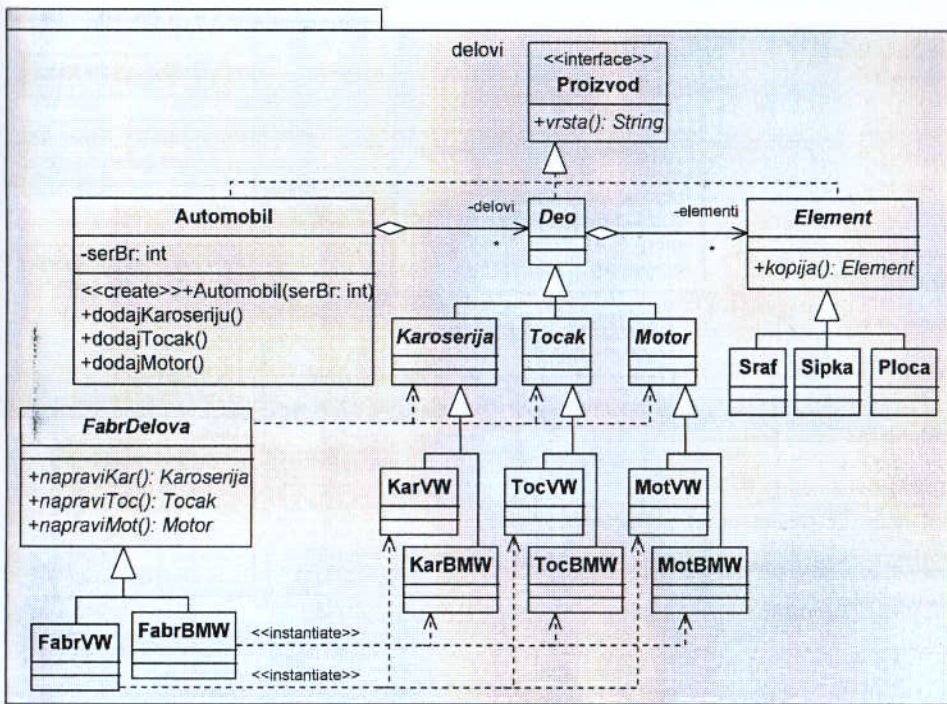
- dijagram klasa razvrstavši ih u pakete (detaljan sadržaj klasa i odnose među klasama na potrebnom broju slika),
- prikaz korišćenih projektnih uzoraka.
- dijagram sekvence za proizvodnju automobila,
- dijagram komponentata stavljajući klase koje čine logičke celine u istu komponentu.

Rešenje:

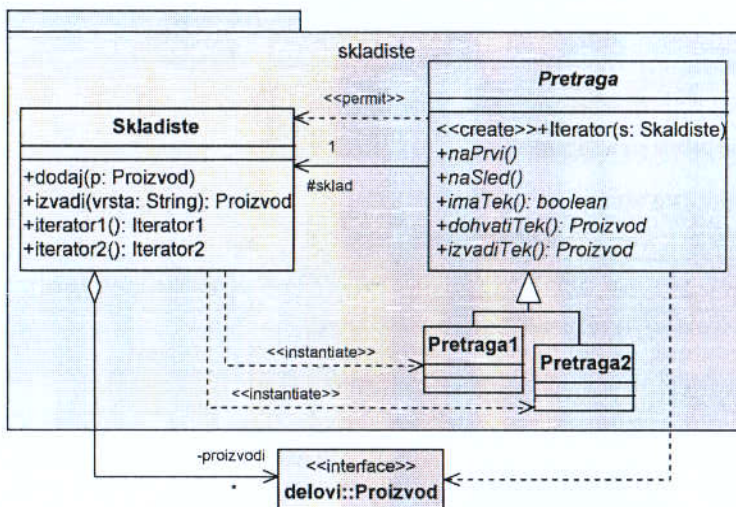
a) Dijagram paketa



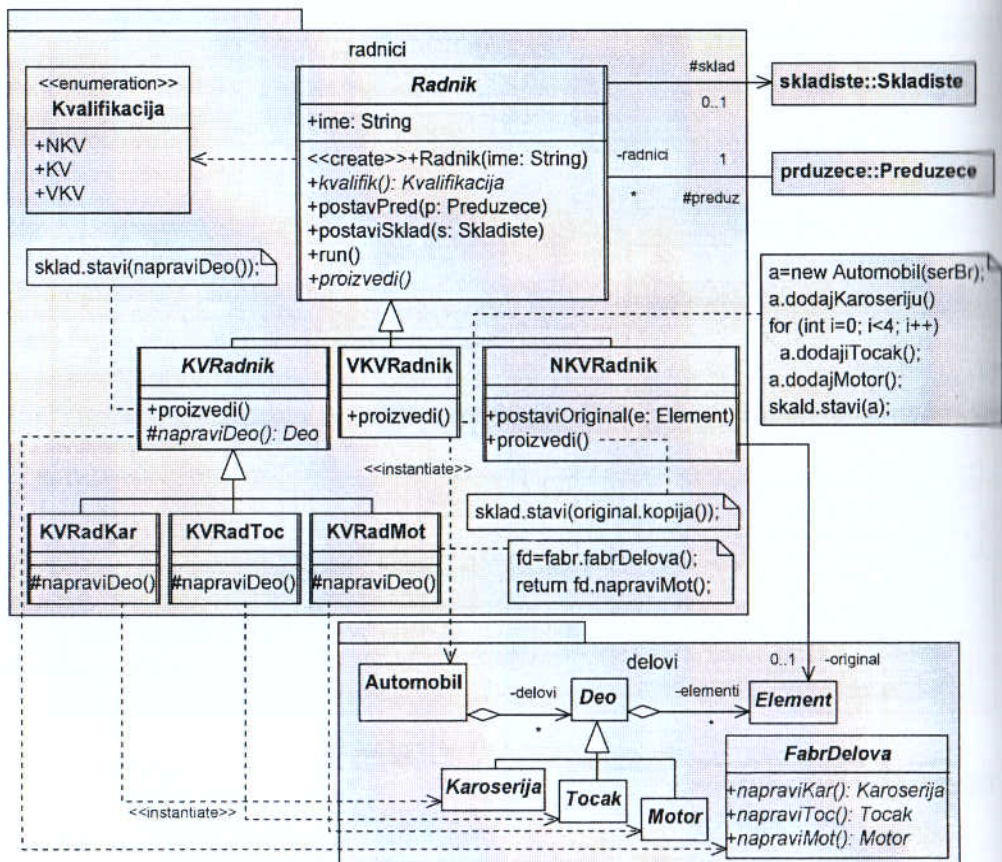
b) Dijagram klasa paketa delovi



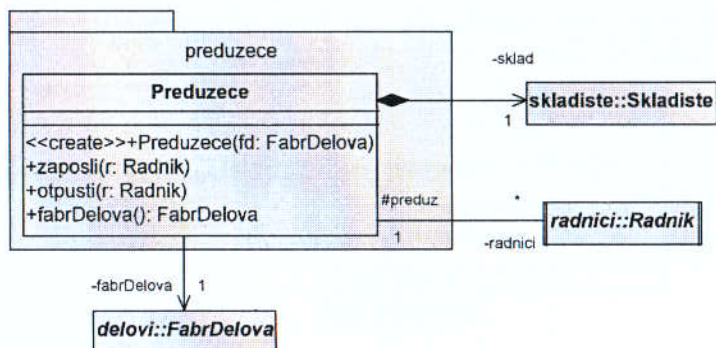
c) Dijagram klasa paketa skadiste



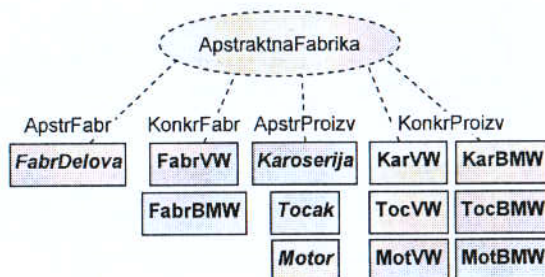
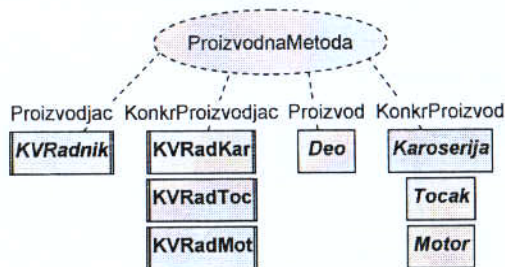
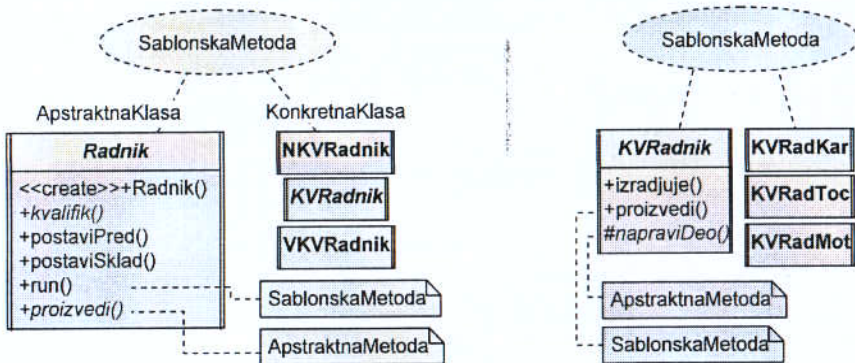
d) Dijagram klasa paketa radnici



e) Dijagram klasa paketa preduzece

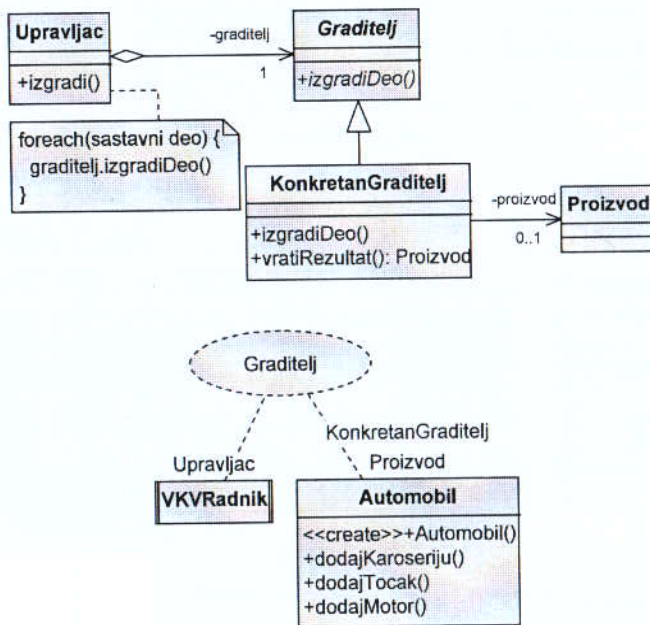


f) Projektni uzorci

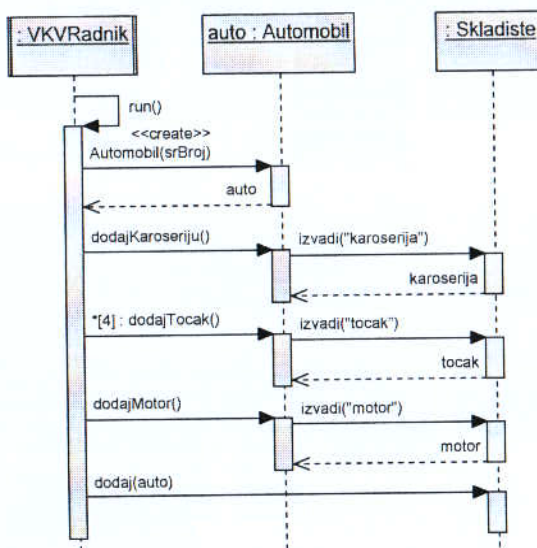


g) Projektni uzorak **Graditelj** (*Builder*)

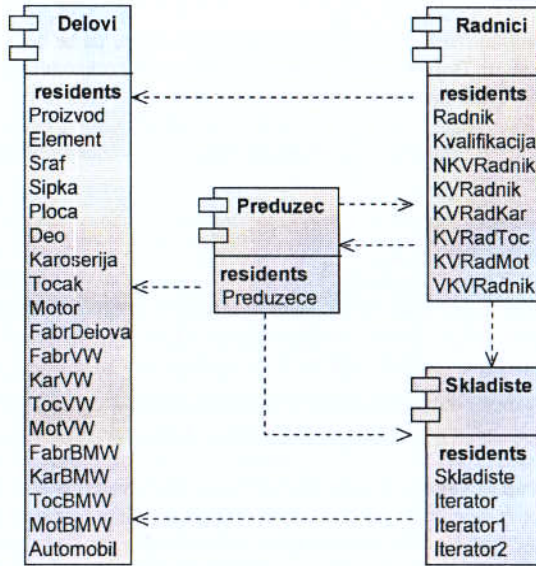
- objektni uzorak stvaranja
- razdvaja izgradnju složenog objekta od njegove reprezentacije
- graditelj konstruiše proizvod korak po korak pod kontrolom upravljača



h) Dijagram sekvence izgradnje automobila



l) Dijagram komponenata



Zadatak 51 Tačka, boja, grafički elementi, crtači, figure, komponente, akteri i program {I, 05.02.2007.}

Tačka u ravni zadaje se pomoću dve celobrojne koordinate koje mogu da se dohvate. Tačka može da se premesti na drugo mesto u ravni. Boja se zadaje pomoću celobrojnih komponenata crvene, zelene i plave boje u opsegu od 0 do 255. Crtač predviđa postavljanje boje pera za crtanje i pravolinijsko pomeranje pera iz trenutnog položaja do zadate tačke bez crtanja i s crtanjem. Ciljna tačka postaje nova trenutna tačka. Konkretni crtači ostvaruju crtanje pod konkretnim operativnim sistemom, koji može biti *Windows* ili *Linux*.

Grafički element predviđa crtanje elementa i pravljenje kopije elementa. Crtanje se ostvaruje pomoću objekta crtača koji može da se postavlja u toku života elementa. Znak je element koji sadrži podatak tipa `char`. Figura je element koji sadrži boju i tačku koja predstavlja položaj gornjeg levog temena. Stvara se bela sa temenom u koordinatnom početku, posle čega ti parametri mogu da se promene. Tekst je figura koja sadrži niz znakova. Stvara se na osnovu podatka tipa `String`. Pravougaonik je figura zadate celobrojne širine i visine. Crtež je pravougaonik koji može da sadrži proizvoljan broj figura.

Komponenta je figura koja može da obradi događaj pritiska mišem. Dugme je komponenta koja sadrži tekst koji čini njegov natpis. Stvara se s praznim natpisom koji posle može da se postavlja. Slika je komponenta koja sadrži jedan crtež. Stvara se s praznim crtežom koji može da se dohvati radi kasnije dorade. Komponente mogu da se uokvire linijom zadate boje. Akter predviđa izvršavanje neke akcije kad se neka komponenta pritisne mišem. Tom prilikom komponenta saopštava akteru koja je komponenta pritisnuta i dostavlja tačku koja predstavlja položaj miša. Akteri mogu da se prijavljuju i odjavljuju kod komponenata radi dobijanja obaveštenja o pritiscima mišem. Promena boje i promena položaja komponente jesu mogući akteri. Ne treba razrađivati detalje izvođenja ovih akcija.

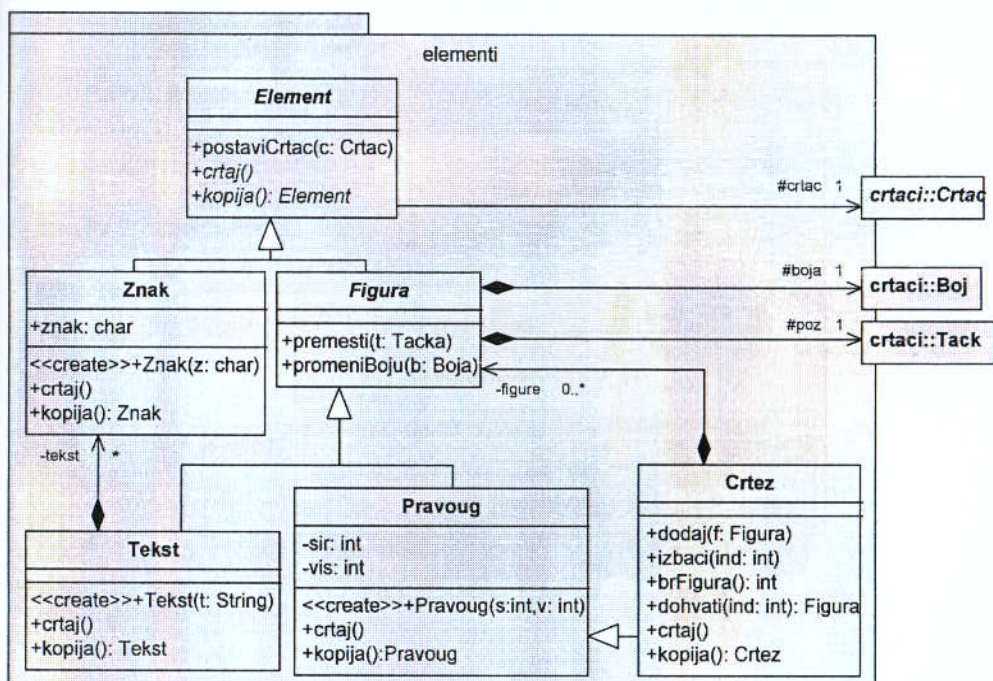
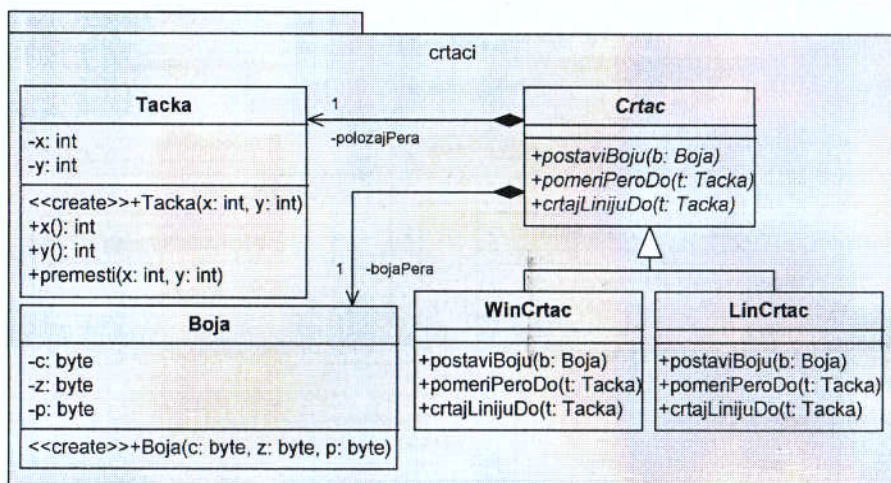
Program obezbeđuje interaktivno crtanje jedne slike korišćenjem crtača koji se zadaje prilikom stvaranja programa. Program podržava početak sastavljanja nove slike, dodavanje figure, izbacivanje figure i vraćanje proizvoljan broj koraka unazad u postupku sastavljanja slike.

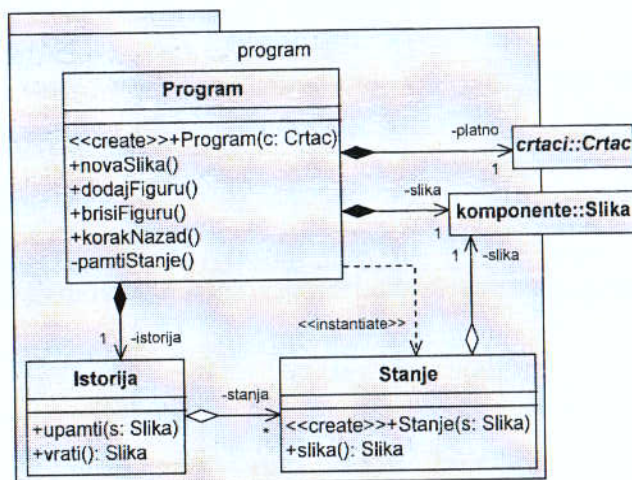
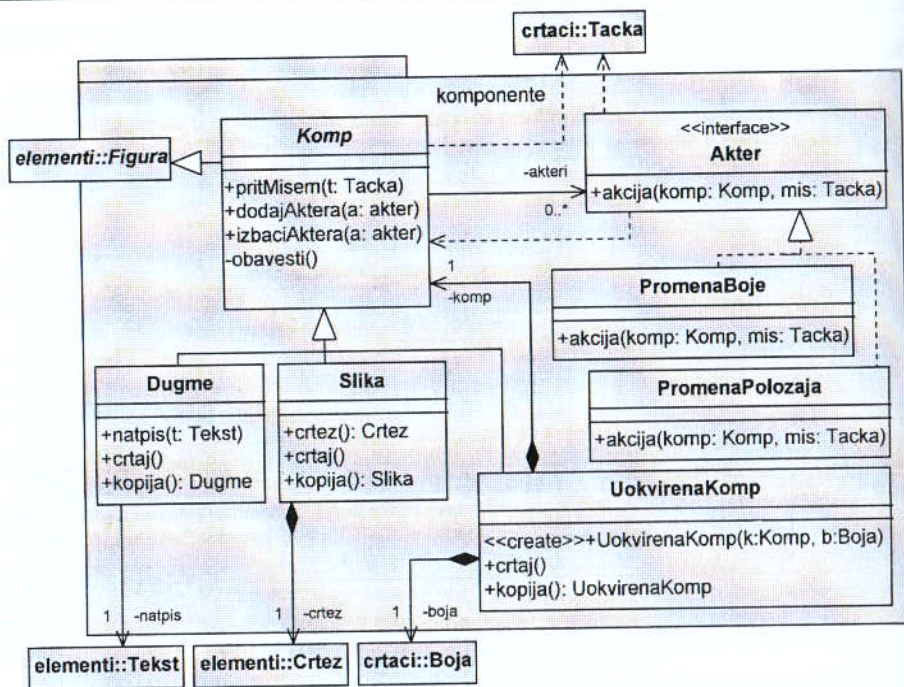
Projektovati na jeziku UML prethodni sistem. Priložiti:

- dijagram klasa razvrstavši ih u pakete (detaljan sadržaj klasa i odnose među klasama na potrebnom broju slika),
- prikaz korišćenih projektnih uzoraka,
- dijagram objekata koji prikazuje jedan crtež koji sadrži jedan uokvireni pravougaonik i jedan tekst od nekoliko znakova,
- dijagram aktivnosti koji prikazuje crtanje uokvirene slike koja sadrži pravougaonike i tekstove,
- dijagram slučajeva korišćenja pri radu s programom,
- dijagram komponenata stavljajući klase koje čine logičke celine u istu komponentu.

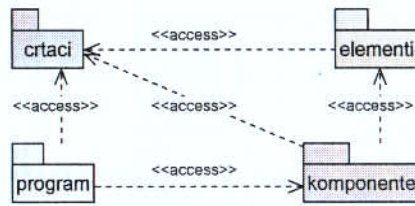
Rešenje:

a) Dijagrami klasa

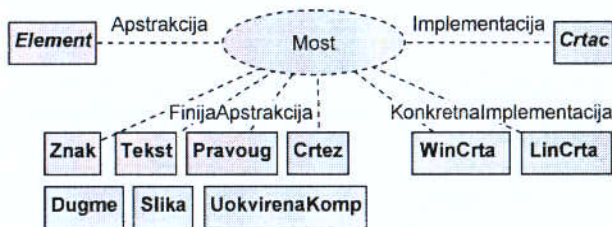
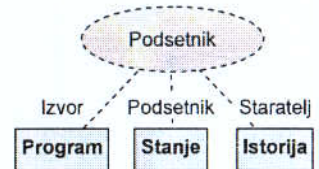
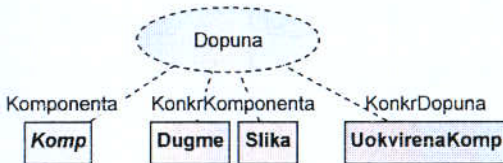
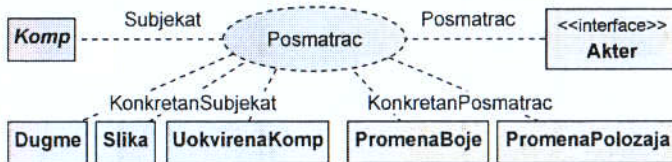
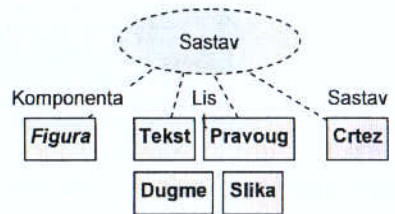
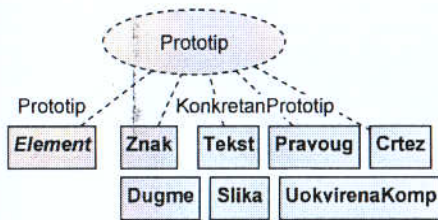




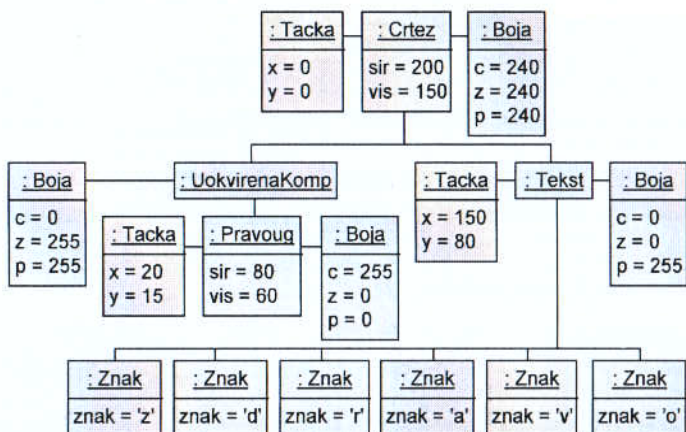
b) Dijagram paketa



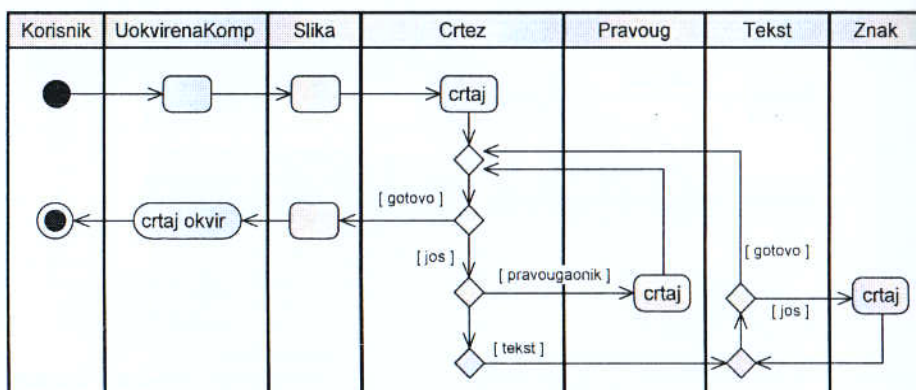
c) Projektni uzorci



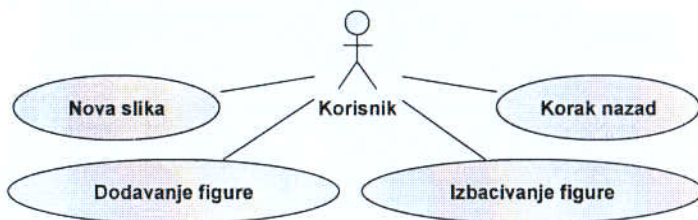
d) Dijagram objekata crteža



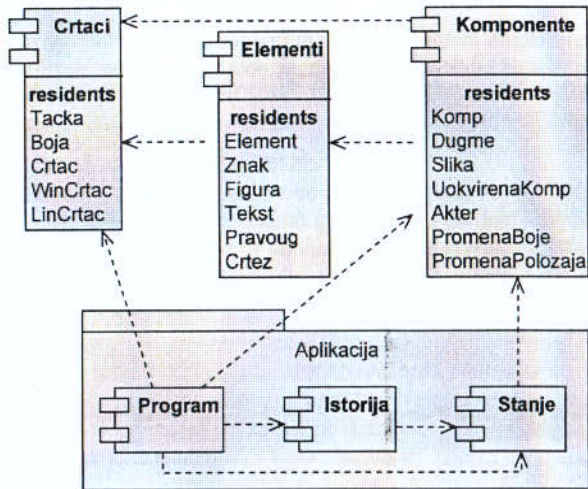
e) Dijagram aktivnosti crtanja uokvirene slike



f) Dijagram slučajeja korišćenja programa



g) Dijagram komponenata



Zadatak 52 Vozila, saobraćajnice i semafor {I, 23.01.2008.}

Aktivno vozilo je opisano snagom motora i trenutnom brzinom. Može da se dohvati trenutna brzina vozila, da se prikaže i da se postavi i dohvati saobraćajnica po kojoj se kreće. Vozilo ima apstraktan model vozila čiji naziv može da se dohvati. Pojedini modeli vozila imaju specifične geometrije i druge osobine. Postoji relativno mali broj različitih modela vozila i veliki broj vozila na saobraćajnicama.

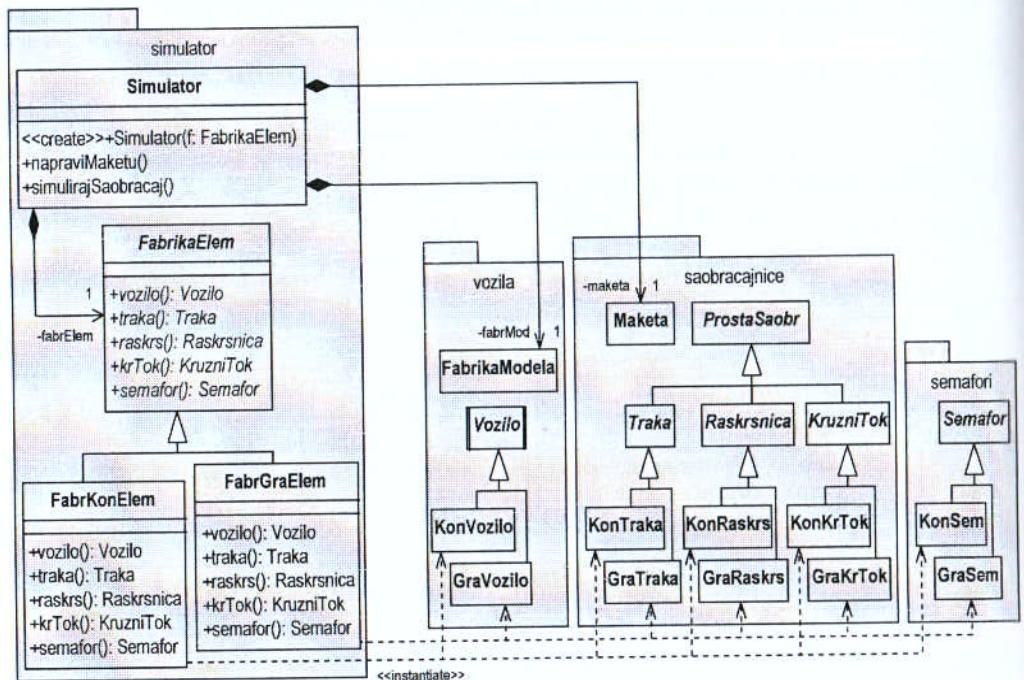
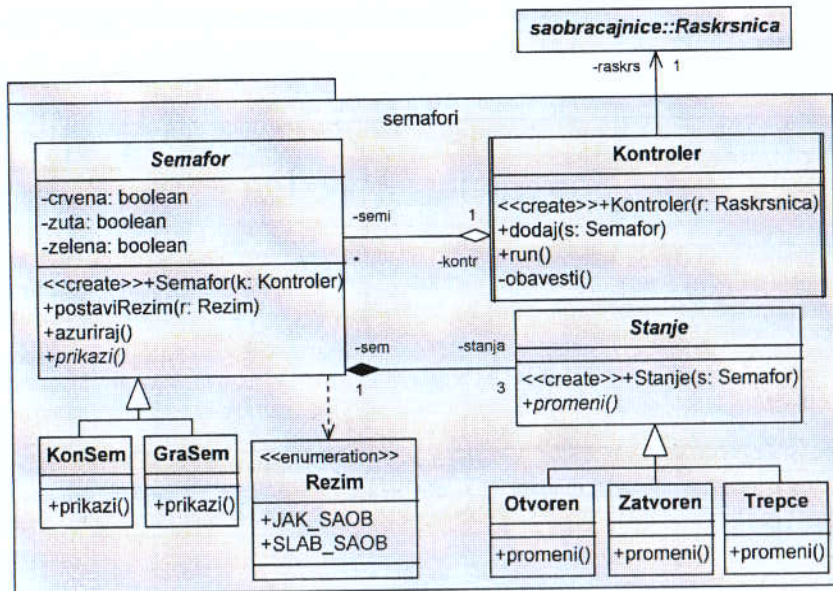
Saobraćajnica može biti povezana sa više drugih saobraćajnica, može da joj se odredi dužina i da se prikaže. Vozila koja se kreću po saobraćajnici mogu da se dodaju i uklone, jedno po jedno. Prosta saobraćajnica ima dužinu. Traka, raskrsnica i kružni tok su proste saobraćajnice. Ulica se sastoji od više traka. Složena saobraćajnica sastoji se od više povezanih saobraćajnica, a dužina joj se određuje kao zbir dužina saobraćajnica od kojih se sastoji. Jedinstvena maketa sadrži jednu složenu saobraćajnicu.

Semafor sadrži crveno, žuto i zeleno svetlo koje može biti uključeno ili ne. Može da radi u režimu jakog i slabog saobraćaja i da bude otvoren, zatvoren ili da trepće. U režimu jakog saobraćaja na zahtev promene stanja iz otvorenog prelazi u zatvoreno i obrnuto. U režimu slabog saobraćaja stalno je u stanju treptanja. Radom semafora na jednoj raskrsnici upravlja aktivni kontroler. Prilikom stvaranja semafor se dodeljuje zaduženom kontroleru. Kontroler šalje svojim semaforima signal za promenu stanja, a svaki semafor određuje svoje naredno stanje na osnovu trenutnog stanja i potrebnih informacija o prohodnim pravcima od kontrolera.

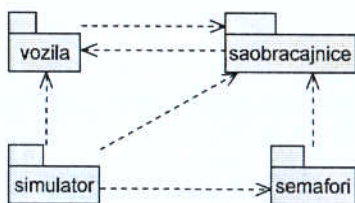
Interaktivan simulator saobraćaja sadrži maketu i može da radi u konzolnom i grafičkom režimu, o čemu se odlučuje prilikom stvaranja simulatora. U konzolnom režimu zahtevi korisnika se primaju preko tastature, a stanja elemenata makete ispisuju u tekstualnom obliku. U grafičkom režimu celokupna komunikacija sa korisnikom odvija se preko grafičke korisničke površi.

Projektovati na jeziku UML prethodni sistem. Priložiti:

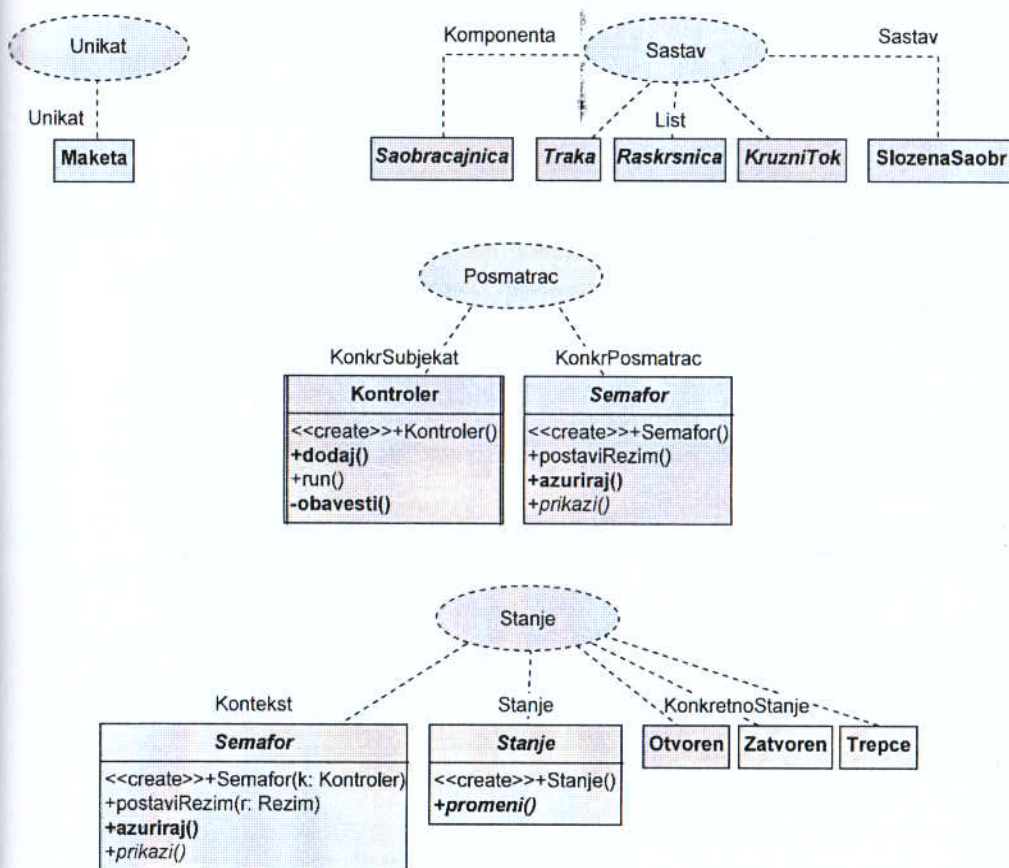
- dijagram klasa razvrstanih u pakete (detaljan sadržaj klasa i odnose među klasama na potrebnom broju slika),
- prikaz korišćenih projektnih uzoraka,
- dijagram sekvence za određivanje ukupne dužine saobraćajnica jedne makete,
- dijagram stanja semafora,
- dijagram komponenata.

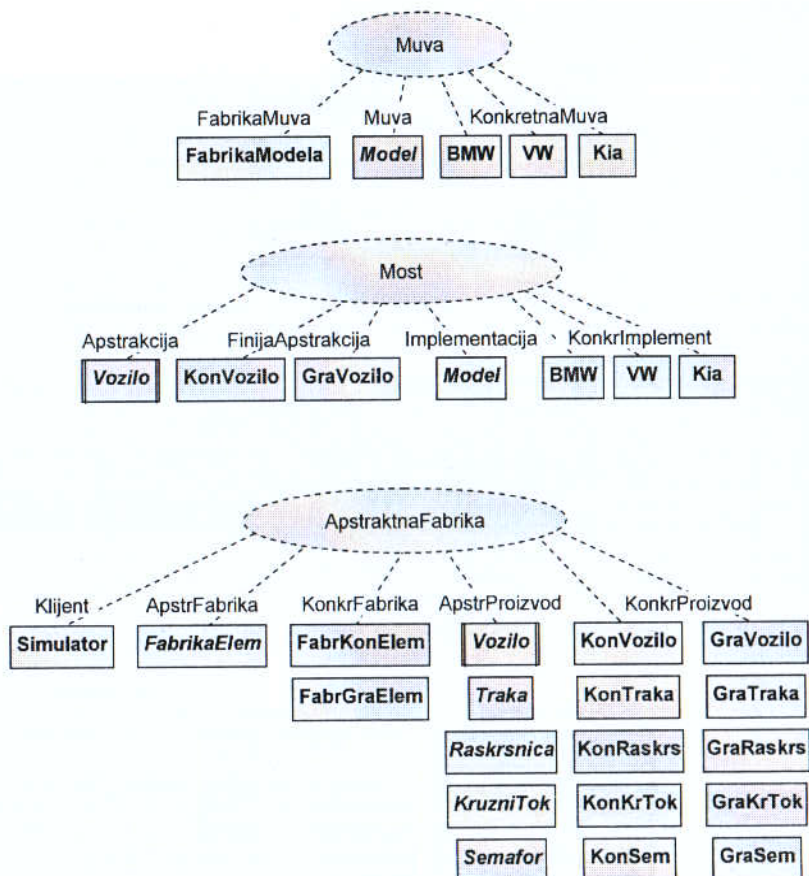


b) Dijagram paketa

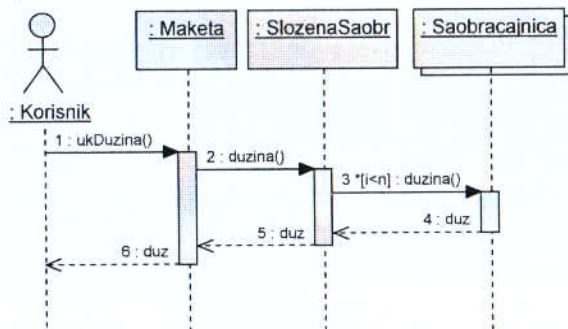


c) Projektni uzorci

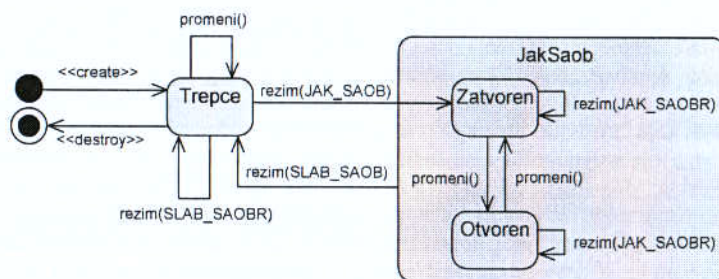




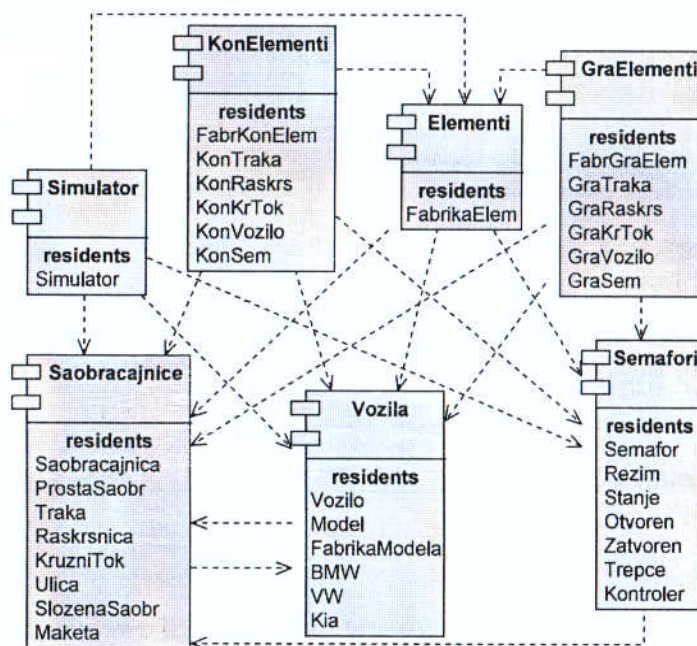
d) Dijagram sekvence određivanja dužine saobraćajnice



e) Dijagram stanja semafora



f) Dijagram komponentata



Zadatak 53 Tačka, boja, duži, telo, scena, crtači, preslikavanje, dugme, radnja i program {I, 28.01.2009.}

Tačka u prostoru zadaje se realnim koordinatama koje mogu da se dohvate. Boja se zadaje celobrojnim komponentama crvene, zelene i plave boje u opsegu od 0 do 255, koje mogu da se dohvate. Broj različitih tačaka i boja je mali. Duž se zadaje dvema krajnjim tačkama i može da se crta zatom bojom. Duž punom linijom i duž isprekidanom linijom su vrste duži. Telo zadate boje sadrži skup tačaka koje čine temena tela i skup duži koje čine ivice. Stvara se prazno posle čega se dodaju temena i ivice. Greška je ako se pokuša dodati ivica čije krajnje tačke nisu sadržane u telu. Scena sadrži tela koja se mogu dodavati i uklanjati jedno po jedno. Mogu da se dohvate sva sadržana tela odjednom, da se sadržaj scene zameni zatom skupom tela i da se scena iscertava na zatom crtaču. Crtanje svih duži vrši se na istom crtaču koji može da se promeni u toku izvršenja programa. Crtač može da nacrtava pravu liniju zatom bojom i stilom (puna linija, isprekidana linija) između zadate dve tačke u trodimenzionalnom prostoru uz projektovanje na dvodimenzionalnu prikaznu površ. Za dobijanje celobrojnih koordinata tačke na prikaznoj površi crtač koristi preslikavanje čije parametre ne treba razmatrati. Ekran i štampač su crtači. Interaktivan program korišćenjem grafičke korisničke površi omogućava obradu jedne scene. Pomoću dugmadi, od programa mogu da se zahtevaju radnje kao što su pravljenje nove scene, dodavanje i uklanjanje po jednog tela, crtanje na ekranu i štampanje na štampaču scene, vraćanje stanja scene proizvoljan broj koraka unazad i završetak rada. Prilikom stvaranja dugmeta definiše se radnja koju će dugme izazvati.

Projektovati na jeziku UML prethodni sistem. Priložiti:

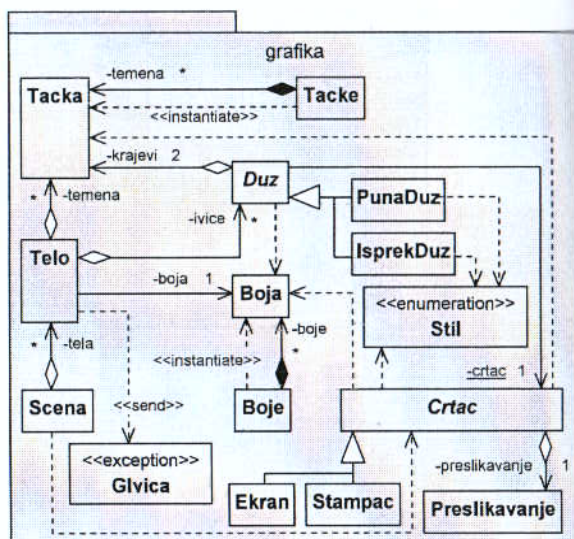
- dijagram klasa razvrstanih u pakete (detaljan sadržaj klasa i odnose među klasama na potrebnom broju slika),
- prikaz korišćenih projektnih uzoraka,
- dijagram slučajeva korišćenja programa,
- dijagram komponenata,
- dijagram aktivnosti pri dodavanju novog tela sceni u programu.

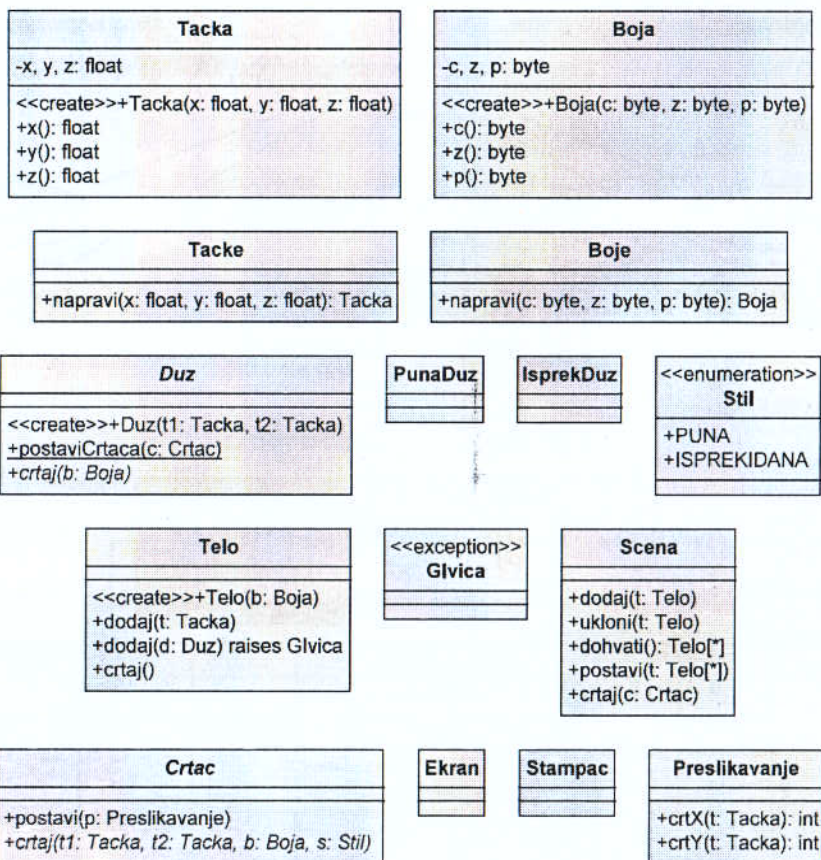
Rešenje:

a) Dijagram paketa

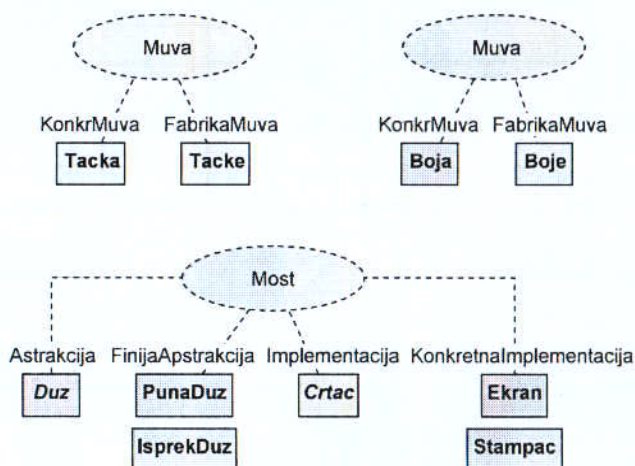


b) Dijagram klasa paketa *grafika*

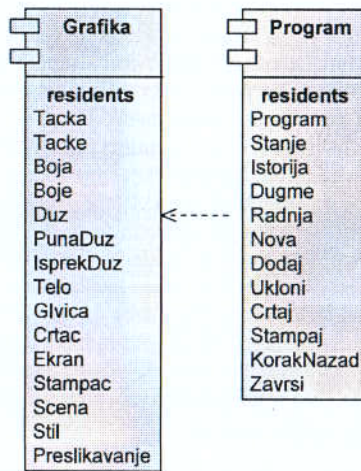




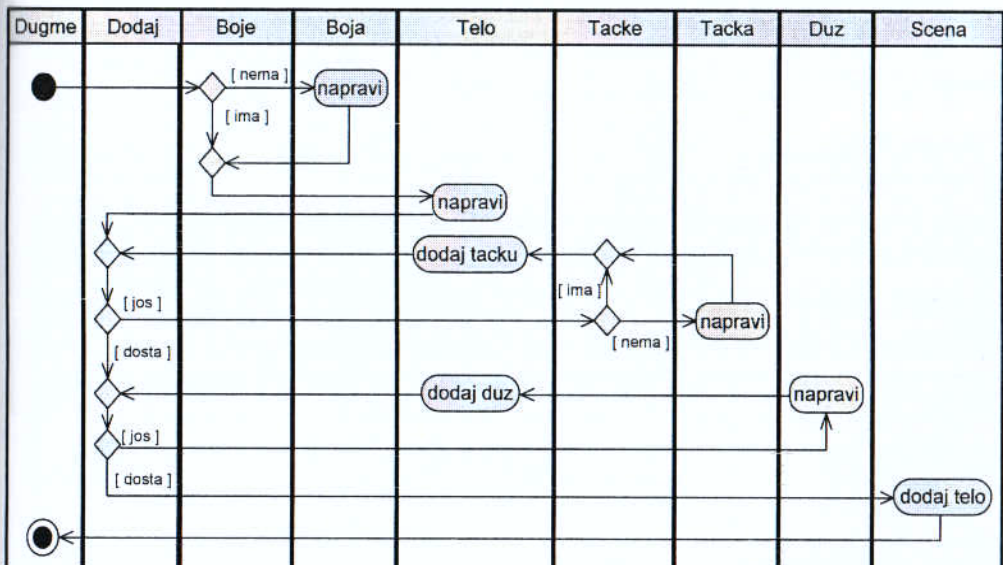
c) Projektni uzorci u paketu grafika



g) Dijagram komponenata



h) Dijagram aktivnosti pri dodavanju novog tela sceni u programu



Zadatak 54 Berza rada [1, 17.01.2012. – Ž.Š.]

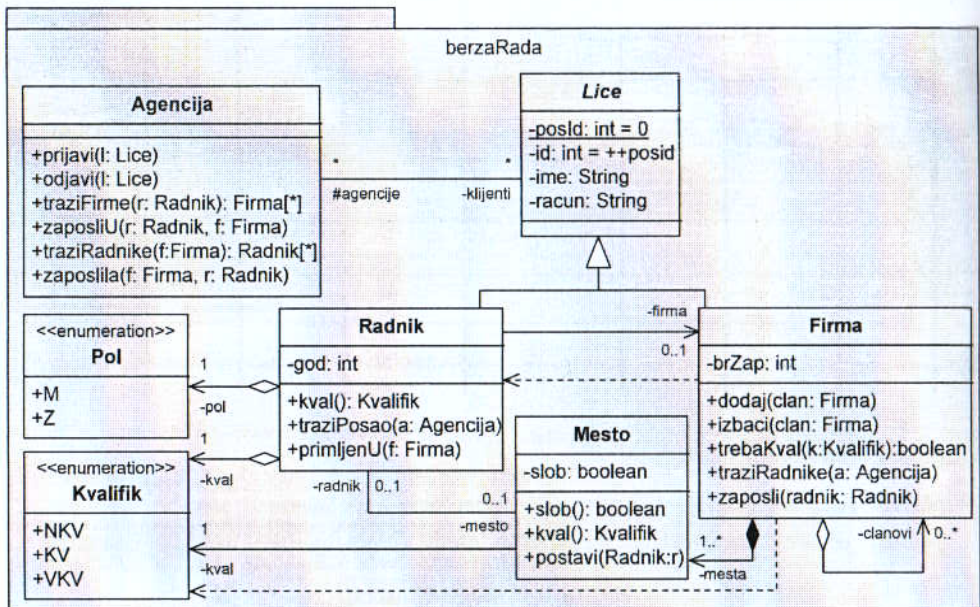
Lice ima jedinstvenu identifikaciju, ime i broj računa. Radnik je (fizičko) lice koje ima pol, starost i kvalifikaciju (nekvalifikovan, kvalifikovan i visokokvalifikovan) i može biti zaposlen na odgovarajućem radnom mestu u nekoj firmi. Radno mesto ima potrebnu kvalifikaciju i može biti slobodno ili popunjeno. Firma je (pravno) lice koje ima broj zaposlenih i radna mesta. Firme se mogu udružiti formirajući nove firme. Agencija za zapošljavanje nudi usluge svojim klijentima (licima) oko uspostavljanja radnog odnosa. Klijenti mogu da se prijave i objave kod agencije radi pružanja usluga. Kada se radnik obrati agenciji, agencija kontaktira prijavljene firme da utvrdi da li neka ima slobodno radno mesto koje odgovara kvalifikaciji radnika. Radniku se vraća zbirka odgovarajućih firmi. Radnik odabere jednu od ponuđenih firmi i posredstvom agencije se zaposli kod te firme. Kada se firma obrati agenciji, agencija kontaktira prijavljene radnike da utvrdi da li postoje radnici koji imaju kvalifikaciju za slobodna radna mesta firme. Firmi vraća zbirku radnika s odgovarajućim kvalifikacijama za slobodna radna mesta. Firma odabere jednog od ponuđenih radnika i posredstvom agencije saopšti radniku da ga je zaposlila. Simulator zapošljavanja omogućava stvaranje agencije i svih lica tako da odgovaraju konzolnom (sa tekstualnim izlazom) ili prozorskom (sa grafičkim izlazom) režimu izvršavanja simulacije.

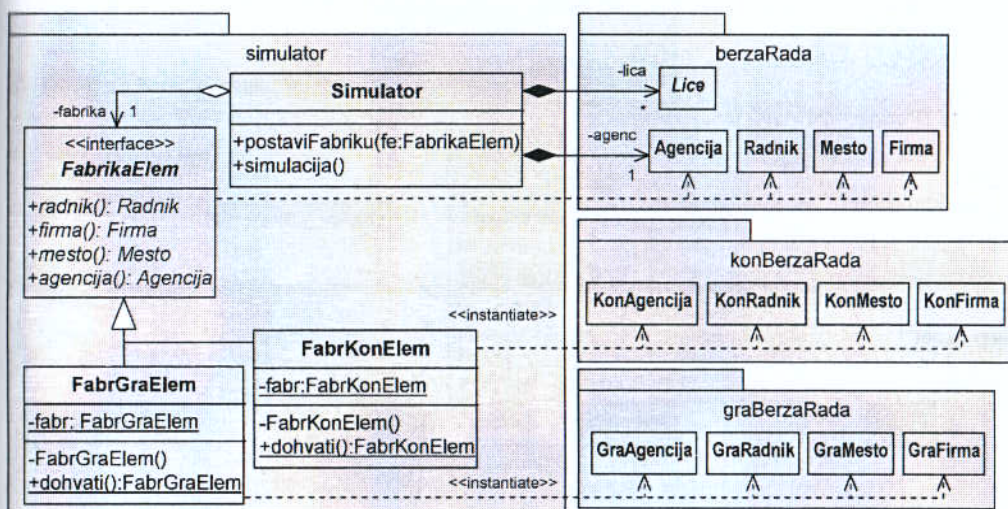
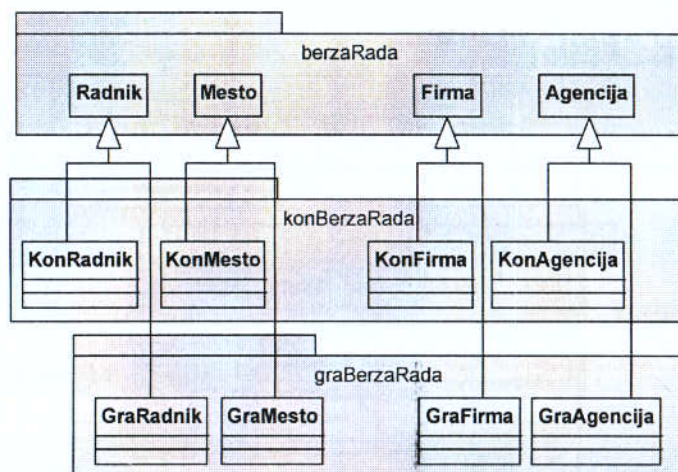
Projektovati na jeziku UML prethodni sistem. Priložiti:

- dijagram klasa razvrstanih u pakete (detaljan sadržaj klasa i odnose među klasama na potrebnom broju slika);
- prikaz korišćenih projektnih uzoraka;
- dijagram aktivnosti s plivačkim stazama za postupak zapošljavanja radnika na njegovu inicijativu;
- dijagram komponenata tako da komponente sadrže klase iz odgovarajućih paketa.

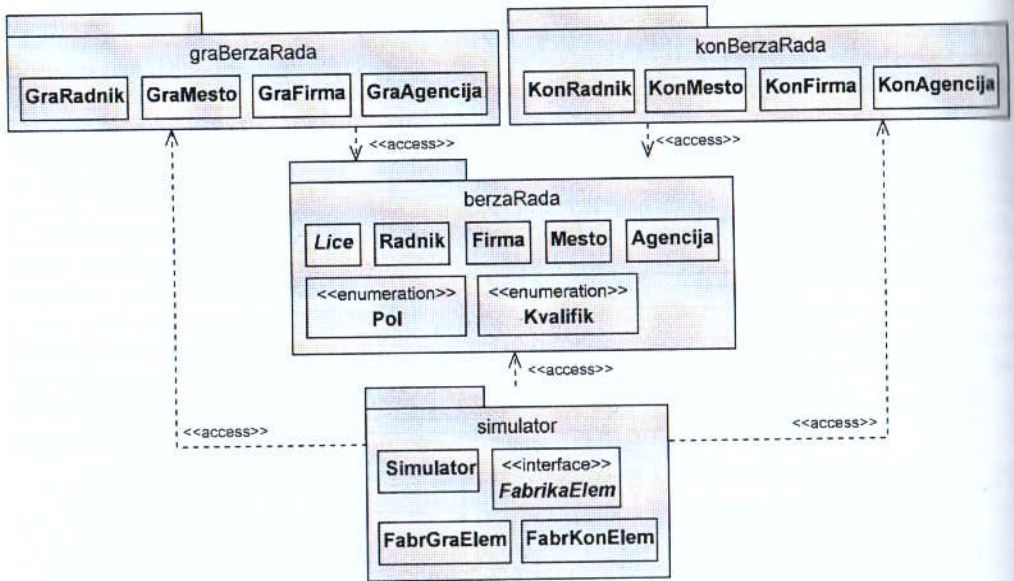
Rešenje:

a) Dijagrami klasa

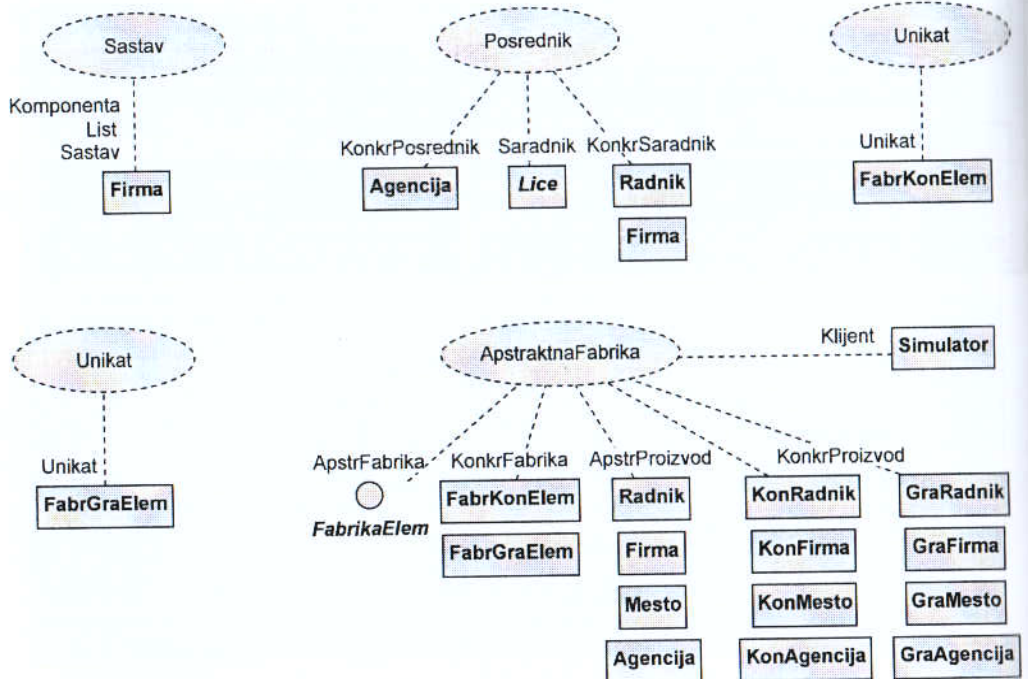




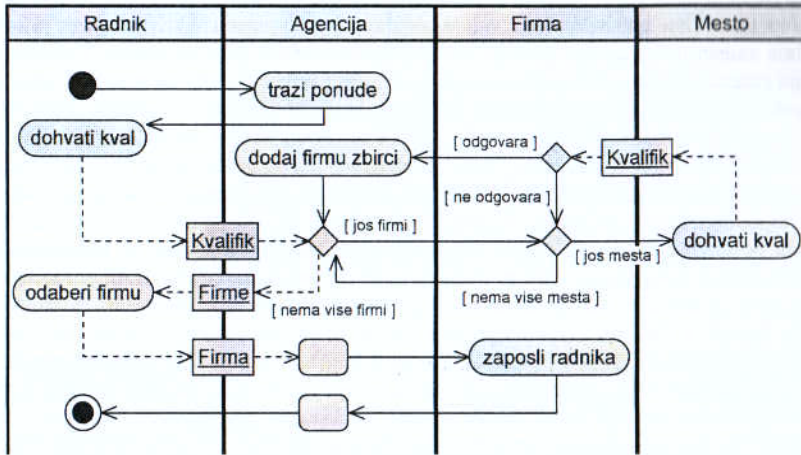
b) Dijagram paketa



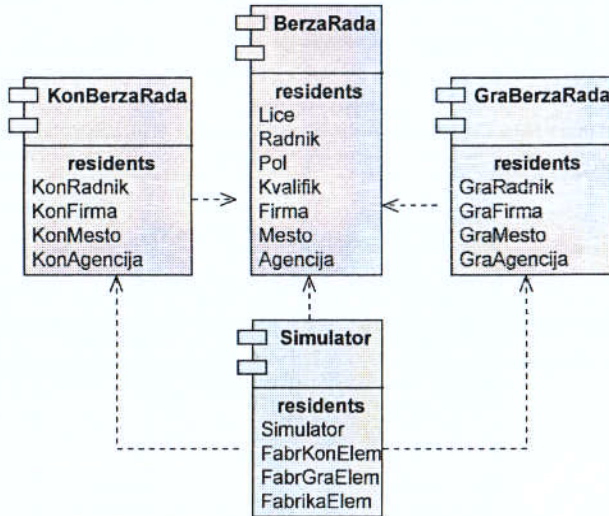
c) Projektni uzorci

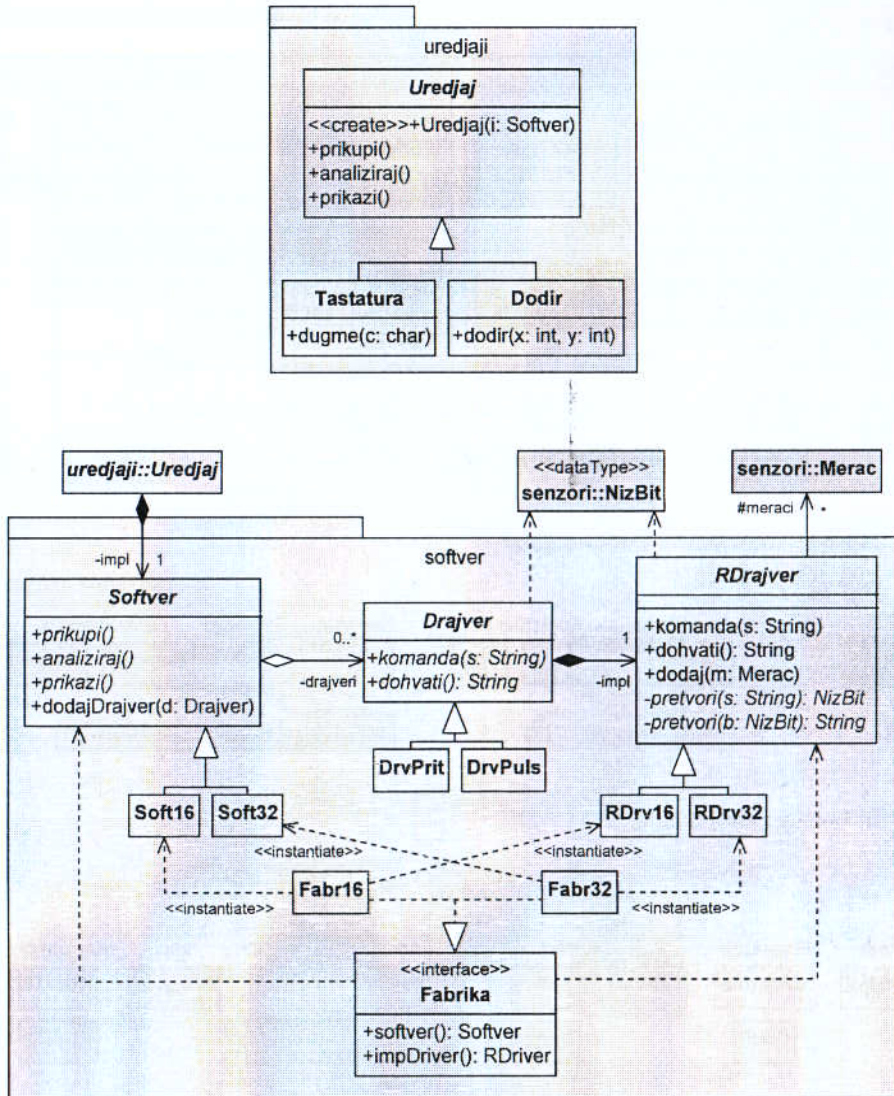


d) Dijagram aktivnosti zapošljavanja radnika

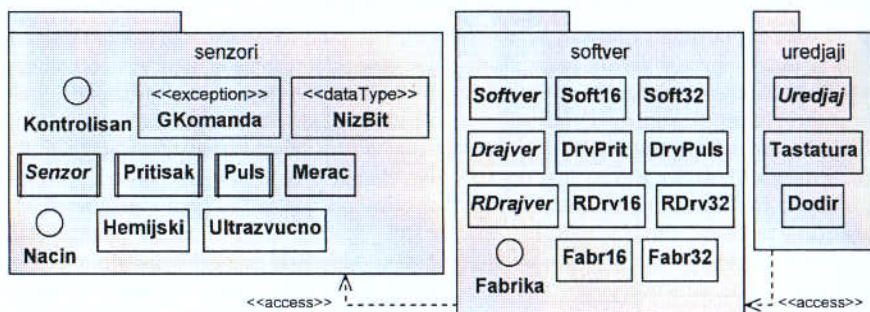


e) Dijagram komponenata

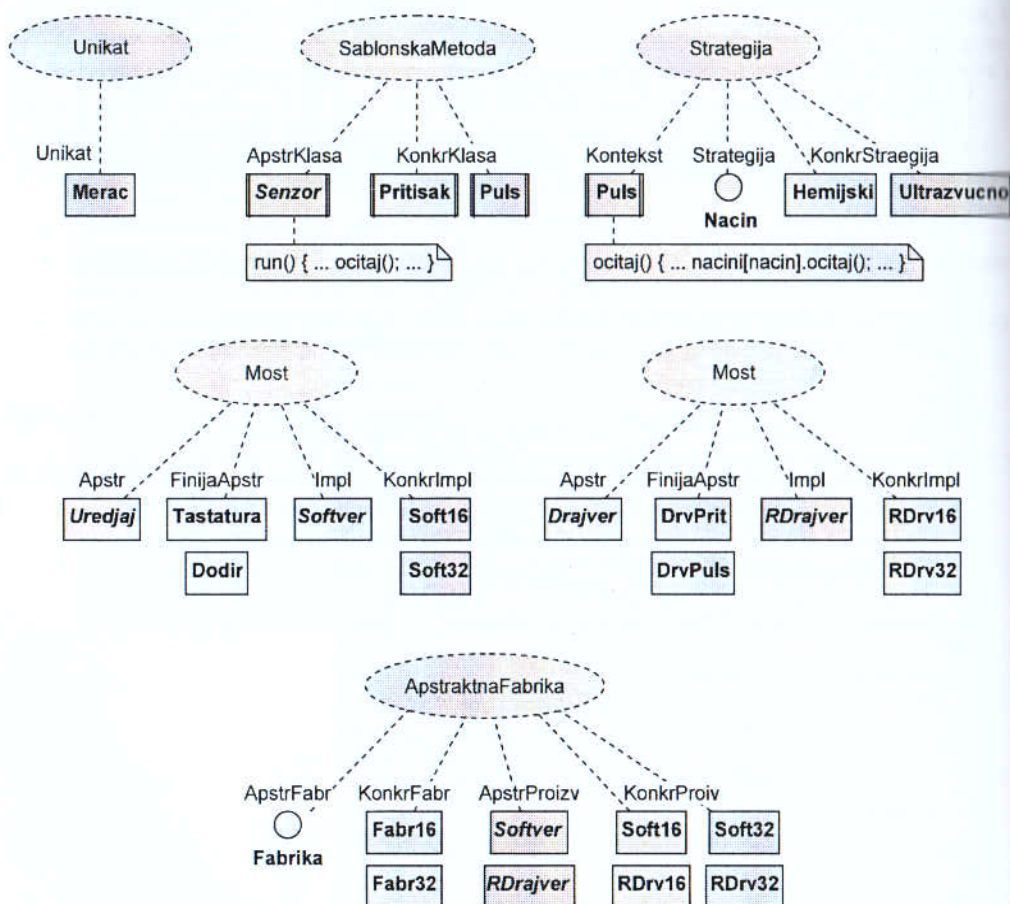




b) Dijagram paketa



c) Projektni uzorci



Ispitni zadaci

Zadatak I Izdavaštvo (I, 21.02.2007.)

Osoba ima ime. Autori, recenzenti i lektori su osobe. Autorsko delo ima autore. Roman, članak i pesma su autorska dela. Izdanje sadrži jedno ili više autorskih dela i ima recenzente i lektora. Knjige i časopisi su izdanja. Zahtev za tiraž jednog izdanja sadrži broj primeraka i stil tog izdanja. Stil izdanja je određen načinom korićenja i vrstom slova. Način korićenja i vrsta slova moraju biti istovremeno generisani za ceo tiraž. Izdavač štampa primerke izdanja. Aktivna odeljenja izdavača su: priprema za štampu, štamparija, knjigoveznica, magacin, transport i knjižare. Izdanje se štampa tako što se zahtev za tiraž preda u odeljenje za pripremu, koje ga prosleđuje odeljenju štamparije, koje ga zatim prosleđuje knjigoveznici, koja smešta primerke izdanja u magacin. Priprema radi prelom izdanja primenjujući odgovarajuću vrstu slova. Štamparija štampa primerke izdanja umnožavanjem šablona izdanja. Knjigoveznica primenjuje razne tehnike korićenja. Svaka tehnika korićenja ima sledeće korake: obrada listova, izrada korica, spajanje. Na primer, za spiralni povez sa mekim koricama listovi se buše, korice se buše i vrši se spajanje spiralom, dok se za tvrdo korićenje listovi ušivaju, na koricama se vrši štampa i korice se lepe. U knjižarama se nalazi veći broj primeraka istog izdanja. Aktivni kupci kupuju u knjižarama primerke izdanja.

Projektovati na jeziku UML model opisanog sistema. Koristiti poznate projektne uzorke. Priložiti dijagram klasa sa naznakom korišćenih projektних uzoraka (na potrebnom broju slika), dijagram sekvence za scenario proizvodnje jednog tiraža izdanja i dijagram stanja za jedno izdanje.

Zadatak II Videoteka {I, 08.03.2007.}

Osobe imaju ime i prezime i jedinstveni broj. Režiseri, glumci i fotografi su osobe. Delo ima naziv. Film je delo koje ima režisera i glumce. Fotografija je delo koje ima fotografa. Apstraktni medijum može sadržati jedno ili više dela. Kaseta i disk su apstraktni medijumi, pri čemu film može biti zapisan na kaseti ili disku, a fotografija samo na disku. Formatirani medijum nije vrsta medijuma već konkretizuje apstraktni medijum, pri čemu formatirana kasete i formatirani disk, kao vrste formatiranih medijuma, konkretizuju odgovarajuće vrste apstraktnih medijuma. Konkretno formatirane kasete mogu biti VHS ili MiniDV. Konkretni formatirani diskovi mogu biti CD ili DVD. Izdanje je formatirani medijum koji sadrži konkretni formatirani medijum u kutiji sa omotom. Videoteka sadrži police na kojima su originali izdanja. Magacin videoteke sadrži police na kojima se čuva dozvoljen broj kopija originala, koje se prave čim original stigne u videoteku. Korisnici su aktivne osobe koje pozajmljuju izdanja iz videoteke. Videoteka ima jedinstvenu evidenciju (čak i u slučaju više videoteke) u kojoj se vode zapisi o pozajmicama. Pri prvoj pozajmici kopije nekog izdanja u evidenciji se formira kompletan zapis o tom izdanju, a pri sledećim pozajmicama samo se u spisak pozajmica dodaje stavka koja sa referiše na taj zapis i na korisnika koji je pozajmio kopiju izdanja. Radnik je aktivna osoba koja iznajmljuje izdanja i vodi evidenciju iznajmljivanja.

Projektovati na jeziku UML model opisanog sistema. Priložiti dijagram klasa raspoređenih u pakete vezane za stvaralaštvo, proizvodnju i korišćenje izdanja, sa naznakom korišćenih projektnih uzoraka (na potrebnom broju slika), dijagram paketa, i dijagram sekvence za životni vek jedne kopije uz zanemarivanje vođenja evidencije.

Zadatak III Medicinski dokumenti {I, 04.07.2007.}

Datum sadrži 3 cela broja (dan, mesec i godina). Osoba sadrži jedinstveni matični broj građana (JMBG), ime i prezime i datum rođenja. Pacijent je osoba koja ima broj zdravstvene legitimacije. Medicinska osoba ima broj radne knjižice i godine staža. Lekar je medicinska osoba koja može imati specijalizaciju. U slučaju promene podataka lekara (npr. prezime ili specijalizacija) ne smeju se promeniti ovi podaci u ranije formiranim dokumentima koji sadrže podatke o lekaru. Medicinski tehničar je medicinska osoba. Medicinsko osoblje je skup medicinskih osoba. Medicinski dokument ima jedinstven celobrojan identifikator, vezan je za konkretnog pacijenta i može da se pretražuje po zadatom uzorku teksta. Medicinski zapis sadrži datum, tekstualni opis i lekara koji beleži zapis. Dijagnoza sadrži šifru bolesti i tekstualni opis bolesti. Šifarnik oboljenja sadrži zbirku mogućih dijagnoza. U slučaju promene opisa neke dijagnoze u šifarniku mora se automatski promeniti njen opis u medicinskim dokumentima u kojima se ona već pojavljuje. Anamneza je medicinski dokument koji sadrži jedan medicinski zapis (datum prijema u bolnicu, tekst koji lekar unosi kroz razgovor sa pacijentom prilikom prijema, lekar koji vrši prijem) i dijagnozu na prijemu. Dekurzus je medicinski dokument koji sadrži zbirku dnevnih medicinskih zapisa o stanju pacijenta za vreme boravka u bolnici. Operativna lista je medicinski dokument koji sadrži jedan medicinski zapis (datum operacije, opis operacije, glavnog lekara koji potpisuje listu), operativnu dijagnozu i medicinsko osoblje koje je obavilo operaciju. Otpusna lista je medicinski dokument koji sadrži jedan medicinski zapis (datum otpusta, tekst predložene terapije, podatak o lekaru koji vrši otpust) i otpusnu dijagnozu. Istorija bolesti je medicinski dokument koji sadrži zbirku medicinskih dokumenata jednog pacijenta koji se kreiraju za vreme jednog lečenja, ali ne može da sadrži druge istorije bolesti. Istorija bolesti može da sadrži samo po jednu anamnezu, dekurzus i otpusnu listu, a operativnih lista može biti proizvoljan broj. Arhiva je zbirka istorija bolesti jedne ustanove.

Projektovati na jeziku UML model opisanog sistema. Koristiti poznate projektne uzorke ili njihove modifikacije, jasno ih naznačiti i diskutovati odstupanja od originalnih uzoraka. Priložiti dijagram klasa raspoređenih u pakete, dijagram paketa, i dijagram sekvence za utvrđivanje broja istorija bolesti u kojima se razlikuju prijemne i otpusne dijagnoze pacijenata koji su lečeni u nekom periodu u datoj ustanovi.

Zadatak IV Testovi (1. 05.09.2007.)

Opis može biti tekst, slika ili složeni opis sastavljen od drugih opisa. Problem ima celobrojnu šifru, realnu težinu i opis problema. Zadatak je problem koji sadrži opis rešenja zadatka. Pitanje je problem koji sadrži skup ponuđenih odgovora od kojih jedan predstavlja tačan odgovor. Odgovor ima oznaku, opis i faktor kojim se množi težina pitanja ukoliko ga odabere ispitanik. Oznaka sadrži jedno slovo. Zbirka problema sadrži proizvoljan broj problema. Moguće je dodavanje, pronalaženje, menjanje i izbacivanje pojedinačnih problema. Baza problema je zbirka svih raspoloživih problema. Test je zbirka odabranih problema koja sadrži datum generisanja. Može da mu se odredi težina kao srednja težina sadržanih problema. Generator testa dobija kao ulazne podatke broj zadataka, broj pitanja i željenu težinu testa, a zatim nekim izabranim postupkom bira iz zbirke odgovarajuće probleme da zadovolji uslove, te sastavlja tekst testa. Program za interaktivno testiranje poziva generator testa koji stvara jedan test, prikazuje tekst testa korisniku, čeka da korisnik preda rešenje, a zatim poziva pregledača koji dodeljuje bodove i prikazuje osvojene bodove korisniku. Rešenje korisnika sadrži odgovore na sva pitanja i opis rešenja za svaki problem. Pregledač ima dve realizacije. U obe realizacije pregledač dohvata svaki odgovor korisnika i odgovarajući problem iz zbirke, stvara ocenjivače problema i upošljava ih. U prvoj realizaciji pregledač razvrstava pitanja i zadatke, stvara dovoljan broj ocenjivača pitanja i ocenjivača zadataka, pa pitanja prosleđuje ocenjivačima pitanja, a zadatke ocenjivačima zadataka. Ocenjivači pitanja i ocenjivači zadataka rade konkurentno. Druga realizacija pregledača stvara samo po jedan primerak ocenjivača pitanja i ocenjivača zadataka i prosleđuje svaki problem ocenjivaču pitanja, a ocenjivač pitanja pregleda pitanja i prosleđuje probleme ocenjivaču zadataka. Ocenjivač pitanja bode odgovore na pitanja množeći težinu pitanja sa faktorom odgovora. Ocenjivač zadataka dohvata rešenje korisnika i opis rešenja odgovarajućeg zadatka iz zbirke, a zatim poredi nekom izabranom tehnikom rešenje korisnika sa opisom rešenja iz zbirke i dodeljuje bodove.

Projektovati na jeziku UML model opisanog sistema. Koristiti poznate projektne uzorke ili njihove modifikacije, jasno ih naznačiti i diskutovati odstupanja od originalnih uzoraka. Priložiti dijagram klasa raspoređenih u pakete, dijagram slučajeva korišćenja sistema i dijagram sekvence za postupak testiranja.

Zadatak V Šalterska služba {1, 29.02.2008.}

Opis službenika ima automatski dodeljen celobrojan identifikator, ime i prezime i funkciju (tekst). Službenici imaju opis i hijerarhijski su organizovani. Upravnik je aktivan službenik koji ima svoje podređene službenike, a sam može biti podređen svom upravniku. Radnik je službenik koji ne može imati podređene službenike. Radnik izvršava neku osnovnu obradu zahteva koji mu je dat. Posebno zaduženim radnicima, bez obzira na vrstu, mogu biti dodata i neka zaduženja. U zavisnosti od dnevnog režima rada (pre podne, posle podne) jedan korak u osnovnoj obradi zahteva, koji je nezavisan od vrste radnika, menja se, dok se dodatna zaduženja ne menjaju. Šalterski radnik, kurir i referent aktivni su radnici koji obavljaju različite osnovne obrade u šalterskoj službi. Šalterska služba ima izvestan broj šaltera na kojima je po jedan šalterski radnik. Iza svakog šaltera postoji red od nekoliko referenata. Neki referenti su u drugim prostorijama. Šalterski radnik prima zahtev klijenta i predaje ga najbližem referentu u pozadini šaltera. Referent u pozadini šaltera, ako je slobodan, rešava zahtev, a ako ne obradi zahtev prosleđuje ga narednom najbližem referentu. Poslednji referent u redu iza šaltera, ukoliko sam ne obradi zahtev, pravi pošiljku u koju stavlja zahtev i koju adresira nekom referentu, a zatim je ubacuje u poštansko sanduče za kurira koji ih raznosi referentima u drugim prostorijama. Dodatno zaduženje radnika "brojača" u šalterskoj službi može biti brojanje zahteva.

Projektovati na jeziku UML model opisanog sistema. Priložiti dijagram klasa smeštenih u pakete koji čine logičke celine sa naznakom korišćenih projektnih uzoraka (na potrebnom broju slika), dijagram objekata (bez navođenja atributa) za šaltersku službu s jednim šalterom i radnicima svih vrsta od interesa u trenutku kad šalterski radnik obrađuje zahtev i kurir nosi zahtev nekom referentu, dijagram sekvence i dijagram aktivnosti obrade jednog zahteva koji biva potpuno obrađen tek od strane referenta u nekoj prostoriji i dijagram komponenata stavljajući klase koje čine logičku celinu u istu komponentu.

Zadatak VI Softverska kompanija {I, 09.07.2008 – Ž.S.}

Softverskom proizvodu može da se dohvati vrsta. Informacioni sistem, aplikacija i softverski modul jesu softverski proizvodi. Informacioni sistem sastoji se od aplikacija, a aplikacije od softverskih modula. Informacioni sistem ima jedinstven celobrojan identifikator. Veb aplikacija, klijent/server aplikacija i desktop aplikacija su aplikacije. Modul za poslovnu logiku, modul za rad s bazom podataka i modul za prikaz podataka su softverski moduli. Kolekcija softverskih rešenja sadrži proizvoljan broj softverskih proizvoda. Može da se dodaje original jednog softverskog proizvoda i da se kopija softverskog proizvoda zadate vrste dohvati iz kolekcije softverskih rešenja. Sadržaj kolekcije softverskih rešenja može da se pretražuje na više načina da bi se pronašla željena vrsta softverskog proizvoda. Neki od načina pretraživanja kolekcije softverskih rešenja jesu pretraživanje po datumu i pretraživanje po veličini. Softverski inženjer ima ime, može da mu se dohvati status, može da se zaposli u nekoj softverskoj kompaniji i može da proizvodi softverske proizvode koje stavlja u zadatak kolekciju softverskih rešenja. Mladi programer, stariji programer i vodeći projektant su statusi softverskih inženjera. Mladi programer kreira softverske module. Stariji programer programira aplikacije pomoću softverskih modula iz kolekcije softverskih rešenja. Specijalizovani stariji programeri programiraju veb aplikacije, klijent/server aplikacije ili desktop aplikacije. Aplikacije se različito programiraju za različite klijente. Smatrati da su moduli dovoljno opšti, odnosno da nisu različiti za različite klijente. Vodeći projektant integriše informacione sisteme tako što povezuje veb aplikacije, klijent/server aplikacije i desktop aplikacije iz kolekcije softverskih rešenja. Softverska kompanija zapošljava softverske inženjere, poseduje kolekciju softverskih rešenja i proizvodi informacione sisteme za različite klijente. Neki od klijenata su banka i kladionica.

Projektovati na jeziku UML model opisanog sistema. Priložiti dijagram klasa sa naznakom korišćenih projektnih uzoraka (na potrebnom broju slika), dijagram sekvence i dijagram aktivnosti za proizvodnju informacionog sistema. Akcijama se smatraju programiranje modula, programiranje aplikacija i integrisanje informacionog sistema.

Zadatak VII Upravljanje nitima i procesima {I, 03.09.2008.}

Akter ima ime, celobrojan identifikator, prioritet i kvant vremena koliko najduže može bez prekida da koristi procesor. Može da bude u jednom od stanja: *nov*, *spreman*, *aktivan*, *blokiran*, *uspavan* i *završen*. U aktivnom stanju izvršava se metoda `radi()`. Jedinstveni dispečer upravlja dodelom procesora spremnim akterima. Izbor aktera može biti na osnovu redosleda dolaženja u stanje *spreman* ili po opadajućim prioritetima. Jedinstveni časovnik po isteku postavljenog kvanta vremena obaveštava dispečera o tome. Akteri mogu da se samouspavljaju u zadatom trajanju, kada se prijavljuju časovniku radi obaveštavanja o protoku vremena. Brigu o isteku vremena spavanja vode sami akteri. Objekat uslova predviđa ispitivanje da li je neki uslov ispunjen. Semafor služi za sinhronizaciju rada aktera. Akter može da zatraži čekanje na semaforu do ispunjavanja zadatog uslova. Ako uslov nije ispunjen akter se blokira. Akteri obaveštavaju semafor da se nešto desilo i da je potrebno da on preispita uslove blokiranih aktera. Tom prilikom se svi akteri u nizu blokiranih aktera, čiji su uslovi ispunjeni, prebacuju u stanje *spreman*. Proces je akter koji izvršava metodu `main()`. Niti su akteri koje stvara i čiji je vlasnik neki proces. Upravljač je proces koji čita naredbe operativnog sistema i stvara procese kojima se realizuju te naredbe. Jedinstveni operativni sistem sadrži dispečera, časovnik, upravljač i izvestan broj procesa.

Projektovati na jeziku UML model opisanog sistema. Priložiti dijagram klasa sa naznakom korišćenih projektnih uzoraka (na potrebnom broju slika), dijagrame sekvence koji opisuju scenarije raspoređivanja aktera i to gubljenja i dobijanja procesora u slučajevima isticanja kvanta vremena, čekanja na semaforu i spavanja i dijagram stanja aktera od stvaranja do završetka.

Zadatak VIII Igre na tabli, šah [I, 17.09.2008.]

Apstraktna aktivna igra sadrži apstraktnu dvodimenzionalnu tablu i apstraktne igrače. Može da se postavi početno stanje igre, da se sprovede igra, kad se igračima ciklički zahteva povlačenje poteza i da se proveriti da li je igra završena, kad se proglašava i pobednik igre. Tabla sadrži polja koja mogu da budu prazna ili da sadrže po jednu apstraktnu figuru. Na tabli može da se postavi figura na prazno polje, da se ukloni figura s nepraznog polja, da se premesti figura s jednog na drugo polje (eventualna figura na određišnom polju se uklanja) i da se prikaže sadržaj table. Apstraktna figura zna kom igraču pripada, može da se premesti s jednog mesta na drugo na tabli, uz proveru ispravnosti poteza i da se prikaže. Apstraktan igrač ima boju, igra na zadatoj tabli i može da vuče poteze. Apstraktan računar-igrač je igrač koji automatski vuče potez. Apstraktan korisnik-igrač je igrač koji ručno vuče potez. Igra može da se odvija preko konzole (tastatura i ekran) ili preko grafičke korisničke površi (GUI), pa postoje konkretne table, figure i korisnici-igrači za oba načina igranja, pri čemu je način igranja parametar igre. Šah je konkretna igra. Šahovske figure (kralj, kraljica, ...) jesu figure od kojih svaka zna svoja pravila kretanja. Računar-šahista je računar-igrač u igri šaha. Korisnik-šahista je korisnik-igrač u igri šaha.

Projektovati na jeziku UML model opisanog sistema. Priložiti dijagram klasa sa naznakom korišćenih projektnih uzoraka (na potrebnom broju slika), dijagram komunikacije (sarađnje) za jednu igru šaha u kojoj učestvuju računar-igrač i korisnik-igrač.

Zadatak IX Prodaja računara {1, 09.07.2008. – Ž.S.}

Računarska komponenta ima model i serijski broj koji mogu da se dohvate. Model je određen vrstom, nazivom i proizvođačem komponenta koji mogu da se dohvate. Procesor, matična ploča, operativna memorija i hard disk su računarske komponente odgovarajućih vrsta. Može da se dohvati vrsta komponente i da se odredi cena komponente pomoću jedinstvenog cenovnika. Cenovnik računarskih komponenta za svaki model sadrži stavku sa cenom tog modela. Cena u stavci može da se dohvati i da se promeni. Cenovnik može da se pregleda po vrsti, po modelu i po ceni komponenta. Proizvođač računarskih komponenta ima naziv i proizvodi računarske komponente koje stavlja u pridruženo skladište. Postoje proizvođači procesora, matičnih ploča, operativnih memorija i diskova. Skladište može da sadrži proizvoljan broj komponenta. Stvara se prazno posle čega se komponente dodaju jedna po jedna. Može da se odredi broj komada zadanog modela u skladištu i da se iz skladišta odjednom izvadi zadati broj komponenta zadanog modela. Greška je ako se u skladištu ne nalazi traženi broj komponenta. Prodavnica računara sadrži skladište računarskih komponenta i zapošljava i otpušta radnike. Radnik ima ime, može da se zaposli u zadatoj prodavnici i da obradi zadati radni nalog. Prodavac, tehničar i dostavljač su radnici. Prodavnica može da nabavi odjednom veći broj komponenta i da prodaje jedan računar na osnovu narudžbenice primljene od datog kupca. Narudžbenica sadrži spisak modela koji treba da se ugrade u računar. Kupac ima ime i može da naruči računar u datoj prodavnici na osnovu date narudžbenice, da plati zadatu sumu i da preuzme računar. Prodaja počinje stvaranjem radnog naloga na osnovu narudžbenice i podatka o kupcu. Nalog se potom prosledi prodavcu koji proveri da li u skladištu postoje sve potrebne računarske komponente. Ako je sve u redu, izračuna cenu računara, naplati od kupca i prosledi nalog tehničaru. Tehničar sastavlja računar i prosledi dostavljaču koji gotov računar isporučuje kupcu.

Projektovati na jeziku UML model opisanog sistema. Koristiti poznate projektne uzorke ili njihove modifikacije, jasno ih naznačiti i diskutovati odstupanja od originalnih uzoraka. Priložiti dijagram klasa raspoređenih u pakete, dijagram sekvence i dijagram aktivnosti za jedan scenario koji obuhvata sve korake kupoprodaje računara.

Zadatak X Čoveče ne ljuti se [1, 15.06.2009.]

Prikazivo je nešto što može da se prikaže. Prikazivi element igre ima pridružen prikazivač. Elementi igre *Čoveče ne ljuti se* su: kocka, mesto, staza, kućica, figura i tabla (videti dole). Za svaki element može da se sastavi tekstualni opis. Apstraktan prikazivač može da prikazuje stanje zadatog elementa. Prikazivač može biti konzolni (za tekstualni prikaz) ili grafički (za prikaz na grafičkoj površi). Kada se elementi prikazuju na grafičkoj površi moguće je odjednom menjati stil prikaza (na primer: 2D ili 3D) svih elemenata. Bačena kocka vraća pseudoslučajan broj od 1 do 6. Mesto može da (privremeno) sadrži figuru. Može da se proverí da li je mesto prazno, da se postavi figura na mesto, da se dohvati i da se ukloni figura s mesta. Polje je mesto koje sadrži i pokazivač na naredno polje. Skretnica je polje koje sadrži i dodatni pokazivač na prvo polje završne kućice. Staza sadrži kružnu listu od zadatog broja polja (n), od kojih je svako ($n/4$)-to polje skretnica. Apstraktna kućica ima jednu od 4 boje (crvena, plava, zelena, žuta) koja može da se dohvati. Polazna kućica je kućica koja sadrži 4 mesta i pokazivač na polje na stazi na kojoj se izlazi iz kućice. Završna kućica je kućica koja sadrži 4 polja. Figura ima boju i pokazivač na mesto na kojem se nalazi. Figura može da se izvede na početno polje staze za datu boju (kada pri bacanju kocke igrač dobije 6) i da se vrati u polaznu kućicu. Može da joj se dohvati boja i mesto na kojem se nalazi. Za zadati broj k od 1 do 6, može da se pomeri na odgovarajuće polje, i to: (1) za k polja unapred na stazi, ako se među njima ne naiđe na polje sa skretnicom na kućicu iste boje; (2) ako se naiđe na skretnicu na kućicu iste boje skreće se na prvo polje završne kućice i napreduje prema kraju kućice (ako je moguće, inače se figura ne pomera); (3) ako se na određišnom polju nalazi figura druge boje ta figura se vraća u polaznu kućicu za njenu boju; (4) ako se na određišnom polju nalazi figura iste boje figura se ne pomera. Tabla sadrži stazu, 4 polazne i 4 završne kućice koje se postavljaju pri stvaranju, a kasnije se mogu dohvatiti. Apstraktan igrač igra na zadatoj tabli zatom kockom, ima boju, 4 figure, polaznu i završnu kućicu (sve u boji igrača). Na početku igrač smešta svoje figure u polaznu kućicu, a kasnije igra poteze tako što baca kocku i vuče figuru na tabli. Ukoliko pri bacanju kocke dobije 6 igrač igra dodatni potez. Rezultat poteza je informacija da li je igrač završio igru. Računar je igrač koji automatski bira i vuče figuru primenjujući znanje određenog nivoa. Početno znanje je nivo znanja kojim se na relativno naivan način bira figura kojom će se igrati. Ekspertsko znanje je znanje izbora figure, koje se zasniva na heurističkim pravilima koja primenjuju dobri igrači. Korisnik je igrač koji interaktivno saopštava figuru koju vuče. Jedinstvena igra *Čoveče ne ljuti se* ima tablu, kocku i niz od 4 igrača, a igra se tako što igrači naizmenično vuku poteze, sve dok jedan igrač ne završi. Posle svakog poteza prikazuju se kocka i tabla.

Projektovati na jeziku UML model opisanog sistema. Koristiti poznate projektne uzorke ili njihove modifikacije, jasno ih naznačiti i diskutovati odstupanja od originalnih uzoraka. Priložiti dijagram klasa raspoređenih u pakete, dijagram sekvence za igru na konzoli s jednim korisnikom i dijagram aktivnosti za pmeranje figure.

Zadatak XI Komunikacioni sistem [I, 12.02.2010., Đ.Đ.]

Servis ima jedinstven, automatski generisan celobrojan identifikator i jednoslovnu oznaku vrste usluge, koji mogu da se dohvate. Može da izvrši uslugu bez parametara, čiji je rezultat niska znakova. Složen servis uključuje nekoliko servisa. Brojač je jednostavan jedinstven servis koji broji koliko puta je od njega zatražena usluga. Slučajan broj je jednostavan, ali nije jedinstven servis, koji prilikom stvaranja dobija "seme", a pri svakom zahtevu usluge izračunava slučajni broj na osnovu prethodnog slučajnog broja. Komunikatoru može da se zahteva uspostavljanje logičke veze (sesije) sa drugim komunikatorom, navodeći vrstu zahtevane usluge u toku sesije, da se zahteva raskid veze, da se postavi upit bez argumenata, čiji je rezultat niska znakova i izda obaveštenje da je sesija uspostavljena, odnosno raskinuta. Greška je ako ne postoji raspoloživ sagovornik za zadatu vrstu servisa prilikom uspostavljanja sesije. Neki komunikatori uvek prijavljuju grešku odbijanja ako se od njih zatraži da uspostave sesiju. Greška je kada se traži raskid sesije ili postavlja upit, ako sesija nije uspostavljena. Veza se sastoji od dva komunikatora koji mogu da se dohvate i između kojih je uspostavljena sesija za komunikaciju. Tabela veza sadrži proizvoljan broj veza. Može da se doda i da se izbaci data veza iz tabele i da se dohvati komunikator koji je sagovornik datom komunikatoru. Zahtev za uspostavu veze sadrži vrstu usluge i podnosioca zahteva. Red zahteva se stvara prazan, nakon čega se zahtevi dodaju i uzimaju po *FIFO* principu, uz čekanje ako je red prazan. Komunikacioni uređaj je komunikator koji je povezan sa centralom (videti dole) koja se zadaje prilikom stvaranja. Kod uspostavljanja sesije, uređaj formira zahtev i traži od centrale da uspostavi sesiju sa proizvoljnim sagovornikom koji može da pruži uslugu odgovarajuće vrste. Greška je ako ne postoji raspoloživ sagovornik za zadatu vrstu usluge. Upit uređaja vraća rezultat upita postavljenom sagovorniku, preko centrale. Sprežni uređaj je komunikator koji služi kao spona između centrale i servisa. Servis se zadaje prilikom stvaranja, a sprežnom uređaju može da se dohvati vrsta pridruženog servisa. Sprežni uređaj prijavljuje grešku kada se od njega zatraži da uspostavi sesiju. Upit sprežnom uređaju nikad ne izaziva grešku i vraća rezultat usluge koju dodeljeni servis izvrši. Aktivna centrala sadrži red zahteva, tabelu veza i niz sprežnih uređaja koji su u početku svi prazni. Centrali mogu da se dodaju servisi jedan po jedan, prilikom čega se za svaki stvara i njegov sprežni uređaj. Centrala može da prihvati zahtev za uspostavljanje sesije i stavi ga u red zahteva. Centrala ciklički: uzima zahtev iz reda zahteva, pronalazi slobodan sprežni uređaj za servis zahtevane vrste usluge, napravi odgovarajuću vezu i stavlja je u tabelu veza, sačeka slučajno vreme i obaveštava komunikatora koji je podneo zahtev da je sesija uspostavljena. Može da se zatraži da centrala prekine sve sesije o čemu obaveštava sve povezane komunikatore. Aktivan korisnik ima komunikacioni uređaj koji se zadaje prilikom stvaranja. Korisnik ciklički: zahteva uspostavljanje sesije za neku vrstu usluge, sačeka da se sesija uspostavi, postavlja nekoliko upita svom uređaju u slučajnim vremenskim razmacima u zadatom intervalu i zatim završava sesiju. Rad korisnika može da se pokrene, zaustavi i završi.

Projektovati na jeziku UML model opisanog sistema. Koristiti poznate projektne uzorke ili njihove modifikacije, jasno ih naznačiti i diskutovati odstupanja od originalnih uzoraka. Priložiti dijagram klasa, dijagram paketa, dijagram svih objekata sa nekoliko korisnika i po jednim servisom svake vrste i dijagram aktivnosti sa plivačkim stazama za dobijanje jedne usluge za vreme sesije.

Zadatak XII Sistem komunikacija za rezervisanje parkinga {I, 31.01.2011., N.K.}

Učesnik u komunikaciji ima adresu (niz znakova) koja može da se dohvatiti. Može da primi ili da pošalje poruku. Jedinstveni imenik sadrži učesnike koji se pojedinačno dodaju i uklanjaju. Može da se dohvatiti učesnik iz imenika na osnovu njegove adrese. Greška je ako se učesnik ne pronađe u imeniku. Poruku čini telo (niz znakova), vreme nastanka (ceo broj) i adrese učesnika koji komuniciraju od kojih je jedan pošiljalac, a drugi primalac. Stvara se prazna, posle čega se pojedini podaci postavljaju i dohvataju pojedinačno. Sagovornik je učesnik u komunikaciji koji ima po jedno sanduče za primljene i za poslate poruke. Poruka se može dodati ili izbaciti iz sandučeta. Iz sandučeta se može dohvatati jedna po jedna poruka prema nekom kriterijumu. Jedan od kriterijuma je vreme nastanka poruke, pri čemu se prvo mogu dohvatati ili najstarije ili najskorije poruke. Grupa učesnika je učesnik u komunikaciji koji sadrži druge učesnike. Moguće je dodati ili ukloniti učesnika iz grupe učesnika. Kada grupa prima poruku, poruka se prosleđuje svim njenim članovima. Zahtev može da se izvrši. Usluga je sagovornik u komunikaciji koji, kada primi poruku, napravi na osnovu nje zahtev i izvrši zahtev. Rezervacija parkinga je zahtev koji sadrži registraciju parkiranog vozila (niz znakova), vreme početka parkiranja (ceo broj), zonu parkiranja i poruku na osnovu koje je napravljena. Zona parkiranja ima svoj broj, najduže trajanje parkiranja i cenu sata parkiranja. Postoji jedinstveni skup od nekoliko zona. Usluga parkiranja je usluga koja na osnovu primljene poruke pravi rezervaciju parkinga i poseduje aktivnu evidenciju parkiranja kojoj se mogu dodavati rezervacije. Rezervacije u evidenciji su uređene po vremenu isteka. Rezervacija parkinga izvršava se tako što se prosleđuje evidenciji parkiranja. Kada se rezervacija prosledi evidenciji, evidencija šalje poruku o započinjanju parkiranja sagovorniku koji je poslao odgovarajuću poruku za rezervaciju. Evidencija čeka trenutak isteka prve rezervacije kada uklanja isteklu rezervaciju i šalje poruku o isteku vremena parkiranja sagovorniku koji odgovara toj rezervaciji parkinga.

Projektovati na jeziku UML model opisanog sistema. Koristiti poznate projektne uzorke ili njihove modifikacije, jasno ih naznačiti i diskutovati odstupanja od originalnih uzoraka. Priložiti dijagram klasa razvrstanih u pakete, dijagram sekvence za scenario rezervisanja parkinga (od trenutka slanja poruke za rezervaciju do evidentiranja rezervacije sa obaveštavanjem o prihvatanju rezervacije) i dijagram aktivnosti aktivne evidencije parkiranja.

Zadatak XIII Veb forum {I, 27.08.2011. – N.K.}

Projektovati na jeziku UML radni okvir za pravljenje veb foruma.

Forum ima naziv i datum stvaranja, može sadržati proizvoljan broj kategorija, a svaka kategorija ima proizvoljan broj tema. Kategorija i tema imaju naziv. Dozvoljeno je da neke teme budu i u više kategorija. Korisnik foruma ima svoj nadimak, može da vrati svoj tip, da čita poruke na forumu i da bude obavešten. Član foruma je korisnik foruma koji ima mejl adresu i može da piše poruke na forumu. Administrator i urednik su članovi foruma. Urednik može da briše poruke u okviru tema i može da premesti poruke iz jedne teme u drugu. Administrator može da pravi/briše teme, dodaje/briše kategorije foruma, dodaje/briše korisnike foruma i može da vrati sve tipove korisnika foruma koje može da napravi. Pravljenje korisnika ne sme da zavisi od konkretnih tipova korisnika foruma i stoga mora da bude imuno i na dodavanje novih tipova korisnika. Grupa je korisnik foruma koji obuhvata druge korisnike. Tema sadrži proizvoljan broj poruka. Poruka ima predmet, sadržaj, vreme slanja, pošiljaoca (član foruma), temu kojoj isključivo pripada, može da se otvori i da joj se izračuna veličina i srednja ocena. Članovi foruma mogu da ocenjuju poruke. Član foruma može da oceni proizvoljno mnogo poruka, a jedna poruka može da bude ocenjena od strane proizvoljno mnogo članova. Sadržaju poruke može da se odredi veličina, može da se učita i da mu se dohvati kratak opis. Sadržaj može biti tekst, slika, video ili kombinovani sadržaj. Kombinovani sadržaj uključuje proizvoljan broj sadržaja. Slika i video se učitavaju samo kada se poruka otvori i taj sadržaj se nalazi u vidnom polju odgovarajućeg prozora. Veličina poruke jednaka je veličini sadržaja u telu. Dohvatanje veličine i opisa slike ili video zapisa ne zahteva učitavanje datih sadržaja. Veličina kombinovanog sadržaja jednaka je zbiru veličina sadržaja od kojih se sastoji. Pitanje je poruka. Odgovor je poruka koja ima informaciju o poruci na koju se odgovara, formirajući tako nit komunikacije. Porukama u okviru teme može se pristupati redom po vremenu slanja ili prema ocenama. Za forum je moguće izgenerisati zbirni izveštaj osnovnih podataka korisnika, izgenerisati izveštaj o ocenama poruka korisnika foruma i druge izveštaje. Korisnik može zahtevati da bude obavešten kada se pojavi neka nova poruka u temi koju izabere. Ukoliko je prijavljen korisnik grupa, grupa prosleđuje obaveštenje svojim članovima.

Projektovati na jeziku UML model opisanog sistema. Priložiti dijagram klasa smeštenih u pakete koji čine logičke celine sa naznakom korišćenih projektnih uzoraka (na potrebnom broju slika). Priložiti dijagram objekata koji prikazuje otvoren odgovor na pitanje koji sadrži tekst, sliku i video, dijagram sekvence koji prikazuje računanje veličine datog odgovora i dijagram aktivnosti koji prikazuje slanje odgovora na odabrano pitanje u okviru proizvoljne teme.

Zadatak XIV Dohvatanje internet stranica (I, 07.02.2012., Ž.Š.)

Zahtev za dohvaćanje internet stranice sadrži adresu stranice i tekstualne parametre, koji mogu da se dohvate. Prikazivač može da prikaže internet stranicu na osnovu zahteva. List (*tab*) je prikazivač koji sadrži niz datoteka koje čine internet stranicu i ima platno na kojem može da prikaže samo jednu internet stranicu, da prikaže prethodno prikazanu stranicu i da prikaže stranicu koja je bila prikazana posle trenutno prikazane. Datoteka sadrži ime datoteke na disku koje se može dohvatiti i omogućava otvaranje i čitanje sadržaja datoteke. Tekst i slika su datoteke. Prozor je prikazivač koji sadrži jedan ili više listova, vodi računa koji je list trenutno aktivan, ima mogućnost za dodavanje i uklanjanje listova. Prozor prikazuje internet stranicu na trenutno aktivnom listu. Postoje dve vrste prozora, listova i platna, jedna vrsta je specijalizovana za prikazivanje stranica na mobilnim uređajima sa relativno malim ekranom, druga vrsta je specijalizovana za prikazivanje na prenosivim i stonim (*desktop*) računarima. List prikazuje internet stranicu tako što prosledi zahtev komunikatoru i kao rezultat dobije niz datoteka na osnovu kojih napravi platno i prikaže ga. Komunikator je jedinstven objekat u sistemu koji može biti povezan ili nepovezan na Internet. Komunikator, kada je povezan na Internet, na osnovu zahteva dohvata internet stranicu kao niz datoteka, a kada je nepovezan dohvata internet stranicu kao (lokalnu) datoteku sa porukom o grešci. Čitač veba (*browser*) je aplikacija koja ima proizvoljan broj prozora sa listovima i jedan komunikator.

Projektovati na jeziku UML model opisanog sistema. Koristiti poznate projektne uzorke ili njihove modifikacije, jasno ih naznačiti i diskutovati odstupanja od originalnih uzoraka. Priložiti dijagram klasa raspoređenih u pakete, dijagram aktivnosti za realizaciju jednog zahteva dostavljenog listu i dijagram komponenta.

Zadatak XV Elektronsko skladište {I, 05.02.2013. – Ž.Š.}

Proizvodu može da se dohvati ime. Elektronski proizvod ima jedinstven identifikator, ime, cenu kupovine i sadržaj (tipa niza bajtova) koji mogu da se dohvale. Može da mu se odredi cena za jedan dan iznajmljivanja. Knjiga je elektronski proizvod koji ima imena autora i cenu iznajmljivanja koji se postavljaju prilikom stvaranja i mogu da se dohvale. Film je elektronski proizvod koji ima cenu kupovine. Cena za iznajmljivanje filma određuje se na osnovu cene kupovine filma i perioda iznajmljivanja. Skladište služi za smeštanje proizvoda. Proizvodi mogu da se dodaju, dohvale i izbace iz skladišta. Može da mu se odredi veličina slobodnog prostora i da mu se promeni veličina. Elektronska prodavnica sadrži skladište elektronskih proizvoda. Stvara se sa praznim skladištem i elektronski proizvodi se dodaju jedan po jedan. Proizvodima zadate vrste u prodavnici može da se pristupa redom po rastućoj ceni i po rastućoj ceni iznajmljivanja. Korisnik može da kupuje, iznajmljuje i vraća proizvode u prodavnicama. Prilikom prve nabavke (kupovine ili iznajmljivanja) korisnik stiže na spisak korisnika prodavnice. Prodavnica može korisniku sa spiska da naplaćuje usluge i da mu šalje obaveštenja o novim proizvodima. Korisnik može da ima svoje globalno skladište (na nekom serveru, koji nije potrebno modelirati) ili lokalno skladište (na nekom svom uređaju, koji takođe nije potrebno modelirati). U globalnom skladištu čuva se virtuelni proizvod koji ima adresu kupljenog ili iznajmljenog proizvoda iz skladišta prodavnice, uz neke podatke o tom proizvodu. Virtuelni proizvod smešta se u globalno skladište prilikom nabavke proizvoda. U lokalnom skladištu čuva se fizička kopija elektronskog proizvoda. Fizičko lice je korisnik koji koristi jedno globalno i proizvoljan broj lokalnih skladišta i kojem se usluge naplaćuju preko kreditne kartice. Kada fizičko lice nabavi nešto u nekoj prodavnici prvi put, od te prodavnice dobija svoje globalno skladište, te se odgovarajući virtuelni proizvod smešta u globalno skladište, a fizička kopija elektronskog proizvoda u tekuće lokalno skladište (na uređaju preko kojeg je korisnik pristupio prodavnici). Fizički korisnik nije stalno priključen na Internet, te prilikom priključenja na Internet prodavnica korisniku šalje proizvode iz njegovog globalnog skladišta, koji su kupljeni ili iznajmljeni u periodu od prethodnog priključenja, preko nekog drugog uređaja. Firma je korisnik koji može od prodavnice da kupi svoje globalno skladište. Stalno je priključena na Internet i nema lokalnih skladišta. Firma ima kredit koji može da se dopunjuje i koji se umanjuje prilikom naplate. Grafički interfejs za korisnika služi za prikazivanje podataka korisnika i za nabavku ili iznajmljivanje proizvoda. Postoje posebni grafički interfejsi za fizičko lice i za firmu. Grafički interfejs za korisnika ima dve varijante: za stone računare i za mobilne računare.

Projektovati na jeziku UML model opisanog sistema. Koristiti poznate projektne uzorke ili njihove modifikacije, jasno ih naznačiti i diskutovati odstupanja od originalnih uzoraka. Priložiti dijagram klasa raspoređenih u pakete, dijagram interakcije za kupovinu elektronskog proizvoda i dijagram komponenata.

Ispitna pitanja

i UML – uvod

1. Šta je UML?
2. Navesti kategorije korisnika jezika UML.
3. Koje gradivne blokove definiše UML i čemu je namenjena svaka vrsta gradivnih blokova?
4. Koje vrste dijagrama za opis statičkih aspekata modela definiše UML?
5. Koje vrste dijagrama za opis dinamičkih aspekata modela definiše UML?
6. Kako se pravi razlika u notaciji apstrakcija i njihovih pojava (instanci) na jeziku UML? Navesti primere.
7. Šta je stereotip u jeziku UML? Koja notacija se koristi za uvođenje novog stereotipa? Navesti primer standardnog stereotipa.

ii UML – dijagrami klasa

8. Navesti preciznu sintaksu operacije klase.
9. Navesti preciznu sintaksu argumenata operacije klase.
10. Navesti simbole za deklarisanje prava pristupa (vidljivost) klasnim članovima na UML dijagramu klasa i značenje svakog simbola.
11. Kako se na dijagramu klasa može označiti: (1) apstraktna operacija, (2) statička operacija, (3) da je klasa Unikat?
12. Koje osobine (*properties*) vezane za ponašanje u konkurentnom okruženju može posedovati UML operacija? Šta koja osobina znači?
13. Kako se modeliraju aktivne klase i objekti? Koja je razlika između niti i procesa?
14. Šta označava i kako se predstavlja asocijacija na dijagramu klasa? Koji su ukrasi asocijacije mogući? Samo navesti nazive i primere.
15. Šta označava relacija asocijacije na dijagramu klasa i kakav je njen odnos sa vezom (*link*) na dijagramu objekata?
16. Šta označava pojam bidirekionalne, a šta pojam binarne asocijacije?
17. Koje vrste sadržanja podržava UML, koje su razlike i koja je notacija?
18. Koje vrednosti za multiplikativnost strana asocijacije su dozvoljene? Dati primere.
19. Koje su moguće multiplikativnosti na strani celine u relaciji kompozicije, a koje u relaciji agregacije?
20. Šta označava tekstualni ukras "#lozinka" na jednom kraju asocijacije i kako se odgovarajuća informacija može implementirati?
21. Šta označava relacija zavisnosti? Navesti najčešće primere zavisnosti između klasa.
22. Koja je osnovna razlika između relacija asocijacije i zavisnosti?
23. Navesti nazive priloženih relacija, njihova značenja i uloge odgovarajućih stvari na njihovim krajevima.



24. Kako se na dijagramima klasa i dijagramima paketa može naznačiti ugnežđenje?
25. Predstaviti dijagramom klasa logičku strukturu sledećeg primera:

```
class A extends B implements I{
    private static int i;
    private C c;
    public F m(D d){ E e; return new F(); }
}
```

iii UML – dijagrami paketa

26. Kojoj grupi "stvari" pripada, čemu je namenjen paket, koji je njegov simbol i šta predstavlja se na dijagramu paketa?
27. Na kojim dijagramima se pojavljuju, kako se obeležavaju i šta označavaju stereotipi `<<access>>` i `<<import>>`?
28. Koje relacije postoje između paketa A i paketa B u sledećim slučajevima: (1) u paketu A može se koristiti uvezeno javno ime X iz paketa B, ali se to ime ne može koristiti u paketu C kvalifikacijom A : : X, (2) u paketu A nalazi se klasa Y izvedena iz B : : X? Nacrtati odgovarajuće dijagrame paketa.
29. Šta označavaju pojmovi javnog i privatnog uvoza paketa i kojim se grafičkim simbolima predstavljaju na jeziku UML?
30. Šta je radni okvir i kako se predstavlja na jeziku UML?
31. Navesti pravilo zajedničkog zatvaranja (*Common Closure Principle*).

iv UML – dijagrami objekata

32. Koji aspekti modela se predstavljaju na dijagramu objekata: (1) apstrakcije/primeri i (2) statički/dinamički? Koji elementi se pojavljuju na ovim dijagramima?
33. Šta predstavlja dijagram objekata, koji elementi se pojavljuju na njemu i kakav je njihov odnos sa odgovarajućim elementima na dijagramu klasa?
34. Koji se ukrasi asocijacije mogu pojaviti na vezama između objekata, a koji ne mogu?
35. Da li ukras multiplikativnosti ima smisla na vezama na dijagramu objekata i zašto?
36. Na koji način dijagram objekata može da se iskoristi za opisivanje dinamičkog ponašanja sistema?

v UML – dijagrami interakcije

37. Koje vrste dijagrama interakcije postoje u jeziku UML 2?
38. U čemu je osnovna razlika između dijagrama sekvence i dijagrama komunikacije?
39. Kako se na dijagramima interakcije predstavljaju asinhronne poruke i kakva notacija se koristi za označavanje rednog broja asinhronne poruke? Navesti primer.
40. Šta označava natpis na strelici poruke na dijagramu komunikacije: G4 . 2?
41. Kakve vrste poruka se pojavljuju na dijagramima sekvence i komunikacije? Priložiti odgovarajuće grafičke UML simbole.
42. Navesti operatore okvira interakcije i njihovo značenje.
43. Šta označavaju fragmenti interakcije sa simbolima `par`, `opt` i `alt`?

vi UML – dijagrami slučajeva korišćenja

44. Šta je slučaj korišćenja (*use case*), a šta scenario?
45. Šta je to akter, koji UML simbol se koristi, na kojim dijagramima se pojavljuje i kojim relacijama se povezuje sa drugim stvarima na datim dijagramima?
46. Kakav je odnos između slučaja korišćenja i scenarija? Zašto?
47. Šta označava puna linija na dijagramima slučajeva korišćenja? Ko su učesnici u odgovarajućoj relaciji i šta oni predstavljaju?
48. Šta označava puna linija na dijagramima slučajeva korišćenja i kada njom mogu biti povezani slučajevi korišćenja?
49. Kako se grafički predstavljaju, na kojoj vrsti dijagrama i šta označavaju relacije uključivanja (*include*) i proširivanja (*extend*).
50. Na kojoj vrsti UML dijagrama se koristi stereotip relacije zavisnosti `<<extend>>` i šta on označava? Priložiti realan primer.

51. Kako se na dijagramu slučajeva korišćenja prikazuju obavezne, a kako opcione (neobavezne) pot-funktionalnosti neke osnovne funkcionalnosti? Priložiti primer.
52. Modelirati odgovarajućim dijagramom slučajeva korišćenja prijem TV programa kablovske televizije sa mogućim prijemom kodiranih kanala.
53. Na kojem UML dijagramu se koristi stereotip zavisnosti <<import>>, a na kojem <<include>> i šta odgovarajuće relacije označavaju?

vii UML – dijagrami aktivnosti

54. Kako se na jeziku UML predstavlja složena aktivnost sa ulaznim i izlaznim parametrima?
55. Nabrojati čvorove i pseudočvorove dijagrama aktivnosti. Za svaki priložiti grafički simbol i navesti šta označava.
56. Navesti notaciju i objasniti razliku između završnog čvora i čvora kraja toka.
57. Kako se na jeziku UML modelira tok objekata na dijagramima aktivnosti?
58. Kako se na dijagramima aktivnosti označavaju paralelni tokovi kontrole? Dati primer.
59. Kako se na dijagramima aktivnosti označavaju slanje signala, prihvatanje događaja, prihvatanje vremenskog događaja, bacanje i prihvatanje (obrada) izuzetka?

viii UML – dijagrami stanja

60. Navesti UML sintaksu prelaza (tranzicija) na dijagramu stanja. Priložiti primer.
61. Koja je osnovna razlika između pojmova samoprelaza (*self-transition*) i unutrašnjeg prelaza stanja na jeziku UML?
62. Na kojim UML dijagramima se pojavljuju i u čemu je razlika između samotranzicije i unutrašnje tranzicije (prelaza)?
63. Na kojim dijagramima se pojavljuju, kako se označavaju i šta predstavljaju "odloženi događaji"?
64. Kako su definisani automati Murovog (*Moor*), odnosno Milijevog (*Mealy*) tipa i koja vrsta automata se predstavlja na dijagramima stanja?
65. Šta označava (navesti termin i značenje) simbol $\odot$, a šta simbol $\odot^*$?
66. Šta su to konkurentna podstanja i kako se modeliraju na jeziku UML?
67. Kako se modelira konkurentnost na dijagramima aktivnosti, a kako na dijagramima stanja?

ix UML – dijagrami klasa, napredniji pojmovi

68. Šta predstavlja klasa asocijacije? Priložiti primer koristeći UML notaciju.
69. Šta je klasa asocijacije, kako se grafički predstavlja na jeziku UML, i sa kojim pojmom u relacionim bazama se može uporediti?
70. Kako se može implementirati klasa asocijacije?
71. Definirati pojam i na primeru ilustrovati n -arnu asocijaciju.
72. Modelirati sledeći iskaz: uspeh nekog studenta u jednoj školskoj godini na jednom fakultetu se opisuje brojem osvojenih kredita i prosečnom ocenom.
73. Šta označava kvalifikator? Priložiti primer.
74. Modelirati sledeći iskaz: na osnovu boje moguće je izdvojiti skup automobila sa nekog parkinga.
75. Šta su zahtevani, a šta realizovani interfejsi i kako se oni označavaju na jeziku UML?
76. Šta je "generalizacioni skup", na kojem UML dijagramu se pojavljuje i koja notacija se koristi? Priložiti primer.
77. Navesti ograničenja generalizacionog skupa uz kratka objašnjenja.

x UML – dijagrami složene strukture

- 78. Šta predstavlja i koji simbol se koristi za konektor na dijagramu složene strukture?
- 79. Šta povezuju, na kojim dijagramima i kako se predstavljaju delegirajući konektori? Skicirati primer.
- 80. Šta je saradnja (*collaboration*)? Navesti UML2 notaciju saradnje sa ulogama, događanja saradnje (*collaboration occurrence*) i priložiti primer saradnje sa događanjima saradnje.

xi UML – dijagrami komponenata

- 81. Šta predstavlja, na kojim dijagramima se pojavljuje i koji je UML simbol za artefakt?
- 82. Kojom relacijom su povezani artefakt i odgovarajuća komponenta? Skicirati primer.

xii UML – dijagrami raspoređivanja

- 83. Definirati pojmove jezika UML: artefakt (*artifact*) i čvor (*node*)? Na kojim dijagramima se pojavljuju? Navesti odgovarajuće simbole.
- 84. Na kojim dijagramima se sreću i koji je odnos između komponenata, čvorova i artefakata?
- 85. Na kojim dijagramima i kako se označavaju uređaji, a kako izvršna okruženja? Skicirati primer.
- 86. Modelirati sledeći iskaz: A.exe koristi B.dll, a oba se izvršavaju na računaru R; B.dll poziva preko Interneta veb servis WS koji se izvršava na serveru S.

xiii UML – dijagrami pregleda interakcije

- 87. Na kojim dijagramima se pojavljuju i šta sadrže čvorovi interakcije i događanja interakcije?
- 88. Kako se prikazuju događanja interakcije na dijagramima pregleda interakcije? Priložiti primer.

xiv Projektni uzorci – Uvod

1. Navesti i obrazložiti načine klasifikacije projektnih uzoraka.
2. Kako se dele projektni uzorci prema kriterijumu namene, a kako prema kriterijumu domena?
3. Kako su klasifikovani projektni uzorci? Definisati klase uzoraka.

xv Projektni uzorci stvaranja

4. Koja je motivacija i kada treba primeniti uzorak Unikata (*Singleton*)? Priložiti dijagram klase koji opisuje uzorak, adekvatnu UML notaciju saradnje za ovaj uzorak i implementaciju uzorka na jezicima Java ili C++.
5. Navesti karakteristične elemente klase uzorka Unikata (*Singleton*) i priložiti klasni dijagram jednog praktičnog primera istoimenog projektnog uzorka.
6. Na koji način se obezbeđuje u uzorku Unikata (*Singleton*) da se ne može napraviti više od jednog objekta odgovarajuće klase?
7. Za koji projektni uzorak je karakteristična metoda polimorfnog kopiranja? Da li bi ta metoda mogla da se pojavi u klasi projektnog uzorka Unikata i zašto?
8. Navesti klasifikaciju i namenu uzorka Fabrički (proizvodni) metod (*Factory Method*). U okviru kojih drugih projektnih uzoraka se često koristi i fabrički metod?
9. Navesti namenu projektnog uzorka Graditelj (*Builder*) i uopštenu klasnu strukturu uzorka.
10. Navesti namenu i priložiti uopšteni klasni dijagram projektnog uzorka Apstraktna fabrika (*Abstract Factory*).

xvi Projektni uzorci strukture

11. Da li je jednostavno kod uzorka Sastav (*Kompozicija*, *Composite*) dodati novu vrstu elemenata? Obrazložiti.
12. Nacrtati dijagram klase koji opisuje klasnu strukturu projektnog uzorka Dekorater (*Dopuna*, *Decorator*).
13. Navesti klasifikaciju i namenu projektnog uzorka Dekorater (*Dopuna*, *Decorator*). Priložiti objektni dijagram primera primene uzorka.
14. Da li klijent treba da bude svestan postojanja objekta dopune (uzorak *Decorator*) i zašto?
15. Koji problem se pojavljuje kod uzorka Dekorater (*Dopuna*, *Decorator*), ako apstraktna komponenta ima atribut?
16. Uporediti po nameni i strukturi (klasnoj i objektnoj) uzorke *Kompozicija* (*Sklop*, *Sastav*, *Composite*) i *Dekorater* (*Omotač*, *Dopuna*, *Decorator*).
17. Navesti klasifikaciju, namenu i klasni dijagram strukture za uzorak Adapter.
18. Koje vrste Adaptera postoje? Priložiti odgovarajuće klasne dijagrame strukture sa naznačenim ulogama klase u projektnom uzorku.
19. Koje vrste Adaptera postoje i šta se naziva dvosmernim adapterom?
20. Navesti klasifikaciju i strukturu projektnog uzorka Muva (*Flyweight*).
21. Navesti uslove kada se uzorak Muva (*Flyweight*) može primeniti.
22. Nacrtati klasni dijagram uzorka Muva (*Flyweight*) i opisati uloge klase u uzorku.
23. Šta označavaju pojmovi unutrašnjeg i spoljašnjeg stanja objekta kod projektnog uzorka Muva (*Flyweight*)? Koje uloge u uzorku pamte odgovarajuća stanja?
24. Koji projektni uzorak bi trebalo koristiti ako bi se veliki broj ponavljanja nekog objekta u nekom kontekstu mogao zameniti referencama na jedan deljeni objekat?
25. Kako je klasifikovan i koja je namena projektnog uzorka Fasada (*Facade*)?
26. Šta je motivacija projektnog uzorka Fasada (*Facade*) i da li projektni uzorak Fasada omogućava direktni pristup metodama objekata podsistema?

27. Navesti klasifikaciju i namenu projektnog uzorka Most (*Bridge*). Nacrtati uopšteni klasni dijagram tog uzorka.
28. Navesti sličnost i razlike između projektnih uzoraka Adapter i Most.

xvii

Projektni uzorci ponašanja

29. Koja je namena i koje su uloge klasa u projektnom uzorku Posmatrač (*Observer*)? Priložiti dijagram klasa koji uopšteno opisuje strukturu uzorka Posmatrač.
30. Priložiti dijagram sekvence koji opisuje osnovni scenario projektnog uzorka Posmatrač (*Observer*).
31. Koji projektni uzorak koristi mehanizam delegirane obrade događaja u paketu AWT jezika Java? Odrediti uloge klasa u datom projektnom uzorku.
32. Koje vrste iteratora prema subjektu koji kontroliše iteraciju postoje i čime se odlikuju?
33. Šta je argument operacije `sledeciElement()` kod varijante uzorka Iterator koja se naziva Kursor (*Cursor*)? U kojoj klasi je realizovana navedena operacija?
34. Koji projektni uzorak nameće takozvani "holivudski princip", na koji način i šta se time dobija?
35. Koja je klasifikacija, namena i kada treba primeniti uzorak Strategija (*Strategy*)? Priložiti dijagram klasa koji opisuje uzorak i adekvatnu UML notaciju saradnje za ovaj uzorak.
36. Priložiti klasni dijagram strukture projektnog uzorka Strategija (*Strategy*).
37. Da li u projektnom uzorku Strategija (*Strategy*) konkretna strategija može da pristupa kontekstu i zašto?
38. Uporediti projektni uzorak Dekorater (Dopuna, *Decorator*) sa Kompzicijom (Sastav, Sklop, *Composition*) i Strategijom (*Strategy*).
39. Kako su klasifikovani, šta variraju i u koje vreme (prevođenja ili izvršenja) projektni uzorci Strategija (*Strategy*) i Šablonski metod (*Template Method*)?
40. Navesti klasifikaciju, namenu i klasnu strukturu sa ulogama, za uzorak Stanje (*State*).
41. Nacrtati uopšteni klasni dijagram uzorka Stanje (*State*). Koji uzorak ima praktično istu strukturu dijagrama klasa?
42. Koje vrste (prema nameni) projektnog uzorka Predstavnik (*Proxy*) postoje?
43. Koji uzorak i na koji način treba iskoristiti za rešavanje odloženog skupog kopiranja?
44. Koji projektni uzorak i u kojoj situaciji se naziva "ambasadorom"?
45. Navesti namenu projektnog uzorka Podsetnik (*Memento*), klasnu strukturu i dijagram sekvence koji opisuje ponašanje uloga u uzorku.
46. Navesti klasifikaciju, namenu i klasnu strukturu uzorka Komanda (*Command*).
47. Da li dodavanje novih komandi zahteva izmenu klasa koje učestvuju u projektnom uzorku Komanda (*Command*) i zašto? Sa kojim uzorcima je povezan uzorak Komanda?
48. Koji uzorak i na koji način pomaže da se realizuje oporavak (*recovery*) od otkaza u kojem je izgubljen trenutno obrađivani dokument, a postoji snimljena njegova prethodna verzija?
49. Navesti klasifikaciju i namenu projektnog uzorka Posrednik (*Mediator*).
50. Opisati motivaciju za korišćenje uzorka Posrednik (*Mediator*) na nekom primeru.
51. U čemu je sličnost i razlika uzoraka Fasada (*Facade*) i Posrednik (*Mediator*)?
52. Kako je klasifikovan i koja je namena projektnog uzorka Lanac odgovornosti (*Chain of Responsibility*)?
53. Koja je namena uzorka Lanac odgovornosti (*Chain of responsibility*) i na koji način se može primeniti zajedno sa uzorkom Kompozicija (Sklop, Sastav, *Composite*)?
54. Navesti namenu i priložiti klasni dijagram (sa naznačenim metodama) koji opisuje opštu strukturu projektnog uzorka Posetilac (*Visitor*).

1. Koje gradivne blokove definiše UML i čemu je namenjena svaka vrsta gradivnih blokova?

Odgovor: UML definiše stvari, relacije i dijagrame. Stvari su namenjene osnovnim apstrakcijama modela, relacije njihovom povezivanju, a dijagrami opisu pojedinih aspekata modela prikazivanjem grupisanih stvari povezanih relacijama.

2. Navesti simbole za deklarisanje prava pristupa (vidljivost) klasnim članovima na UML dijagramu klasa i značenje svakog simbola.

Odgovor: + javni, - privatni, # zaštićeni, ~ paketski.

3. Koje vrste sadržanja podržava UML, koje su razlike i koja je notacija?

Odgovor: UML podržava sadržanje tipa agregacije i kompozicije. Agregacija predstavlja odnos celina-deo u kojem celina nije odgovorna za životni vek dela, deo može postojati i bez celine i deo može biti sastavni deo više celina, dok kompozicija predstavlja odnos celina-deo u kojem je celina odgovorna za životni vek dela, deo ne može postojati bez celine i samo jedna celina može sadržati dati deo. Notacija:



4. Navesti klasifikaciju i namenu uzorka Fabrički (proizvodni) metod (*Factory Method*). U okviru kojih drugih projektnih uzoraka se često koristi i fabrički metod?

Odgovor: Fabrički metod je klasni uzorak stvaranja, a namena mu je delegiranje stvaranja objekta konkretnih klasa potklasi, jer se u apstraktnoj klasi fabrike ne zna za konkretan tip proizvoda. Koristi se često u okviru Šablonskog metoda i Apstraktne fabrike.

5. Da li klijent treba da bude svestan postojanja objekta dopune (uzorak Decorator) i zašto?

Odgovor: Ne, klijent ne treba da bude svesan postojanja objekta dopune, jer dopuna pruža isti interfejs klijentu kao i dopunjeni (dekorisani) objekat. Razlog je što se na taj način objekat može dopunjavati proizvoljnim brojem dopuna, a da klijent o tome ne zna ništa.