BSP metodologija

1. Dati primjer matrice ORGANIZACIJE/PROCESI (mogu biti i druge matrice PROCESI/KLASE PODATAKA, APLIKACIJE/ORGANIZACIJA, APLIKACIJE/PROCESI, APLIKACIJE/KLASE PODATAKA).
2. Dati prikaz moguće tabele izlaza intervjua.
3. Data je matrica PROCESI/KLASE PODATAKA

Klase podataka⇒ Procesi ⇓	A	B	C	D	E
Pr1		C	U		U
Pr2			C		U
Pr3	C	U	U		U
Pr4	U				C
Pr5	U	U			C
Pr6	U		U	C	

Procesi Pr1 i Pr2 pripadaju prvoj fazi životnog ciklusa, dok Pr3, Pr4 i Pr5 pripadaju drugoj fazi dok Pr6 pripada trećoj fazi. Preurediti matricu i definisati poslovne podsisteme.

Rješenje:
Klase podataka treba preurediti tako da se prvo prikazuju one koje koriste podatke za prve procese u okviru životnog ciklusa.

Klase podataka⇒ Procesi ⇓	C	E	B	A	D
Pr1	U	U	C		
Pr2	C	U			
Pr3	U	U	U	C	
Pr4		C		U	
Pr5		C	U	U	
Pr6	U			U	C

Predmetna tabela predstavlja jedno moguće rješenje navedenog problema. Svi U-ovi koji su van podsistema su kandidati da se mogu pridružiti nekom od susjednih podsistema ali se prava analiza može obaviti samo posmatranjem fizičkih karakteristika navedenog sistema "uživo".

4. Kreirati matricu ORGANIZACIJE/PROCESI. Organizacije su Or1, Or2, Or3. Procesi su A1, A2 i A3 iz grupe A, B1 i B2 iz grupe B. Organizacija Or1 je glavna u procesu A1, uključena u procese A2 i B1. Organizacija Or2 je glavna u procesu A3 i djelimično uključena u proces B2. Organizacija Or3 je djelimično uključena u proces B2. (Kao zadatak može doći da se na osnovu zadatih podataka kreira bilo koja matrica PROCESI/KLASE PODATAKA, APLIKACIJE/ORGANIZACIJE, APLIKACIJE/PROCESI, APLIKACIJE/KLASE PODATAKA).

Procesi⇒ Klase podataka ⇓	A			B	
	A1	A2	A3	B1	B2
Or1	X	/		/	
Or2			X		z
Or3					z

2. (a) Kreirati matricu PROCESI-KLASE PODATAKA. Procesi su Pr1 (koristi klase podataka A, B i C a produkuje klasu podataka D), Pr2 (koristi klasu C i D a produkuje klasu E) i Pr3 (koristi klase D i E a proizvodi klasu F). Kreirati matricu PROCESI-APLIKACIJE. Procesi su ponovo Pr1 (koristi aplikaciju APL1 i planira je zadržati kao i aplikaciju APL2 koju ne planira zadržati), Pr2 (koristi aplikaciju APL2 koju planira i zadržati) kao i proces Pr3 (koristi aplikaciju APL1 koju ne planira zadržati dok se planira uvođenje aplikacije APL3).
(b) Kreirati matricu APLIKACIJE-KLASE PODATAKA za planirano stanje na osnovu prethodno kreiranih matrica.

Rješenje:

Klase pod. Procesi	A	B	C	D	E	F
Pr1	U	U	U	C		
Pr2			U	U	C	
Pr3				U	U	C

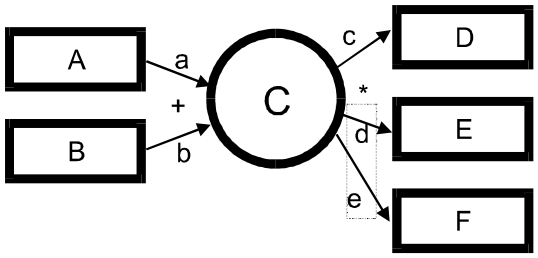
Procesi Aplikacije	Pr1	Pr2	Pr3
APL1	CP		C
APL2	C	CP	
APL3			P

Sada treba kreirati matricu APLIKACIJE-KLASE PODATAKA za planirano stanje. Kako se to radi? Pogleda se sa kojom aplikacijom je vezan koji proces u planiranom stanju. Npr. proces Pr1 je vezan sa aplikacijom APL1 a ovaj proces koristi kao ulaze klase podataka A, B i C. Dakle zaključujemo da je za aplikaciju APL1 potrebno obezbjediti klase podataka A, B i C. Slično se utvrđuje i za ostale aplikacije.

Klase pod. Aplikacije	A	B	C	D	E	F
APL1	X	X	X			
APL2			X	X		
APL3				X	X	

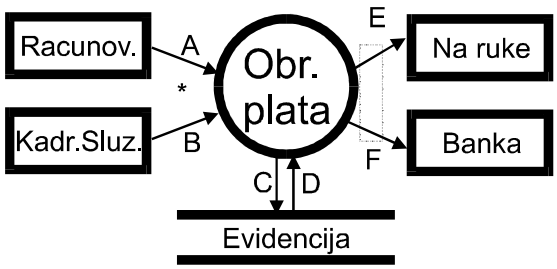
SSA

1. Ilustrovati prikaze osnovnih elementata DTP-a.
2. Ilustrovati mogući izgled formulara rječnika podataka (za tokove podataka, elementarne podatke, skladišta podataka, primitivni proces).
3. Dat je dijagram toka podataka protumačiti njegovo značenje.



Iz izvora podataka A i B u proces C stižu tokovi podataka a i b i to a ili b (može samo a, samo b ili a i b). Rezultat obrade se preko toka c (uvijek prisutan) vodi u uvir D, dok je aktivan i samo jedan od tokova d i e koji vode podatke ka uvirima E i F.

4. Proces obrade plata dobija podatke iz računovodstva preko toka podataka A kao i iz kadrovske službe tok podataka B. Proces komunicira sa skladištem (Evidencija) preko tokova podataka C i D. Za jednog radnika se rezultat prosljeđuje ili "na ruke" preko toka podataka E, ili na žiro-račun u Banku preko toka podataka F (ne može na oba mjesta). Kreirati DTP ovog procesa.

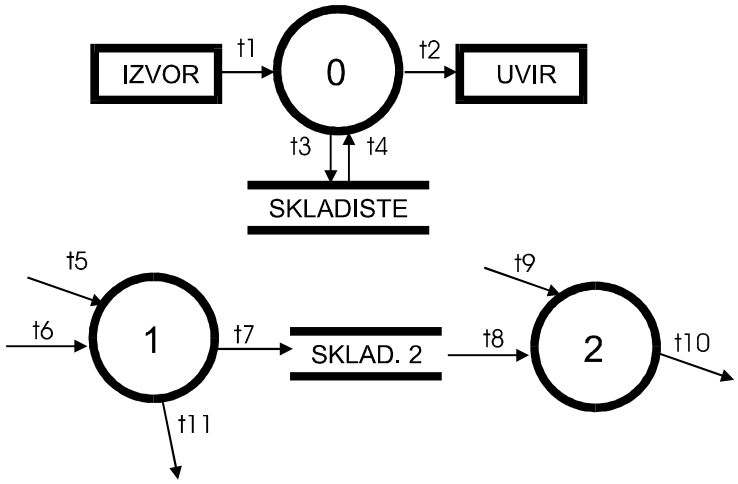


5. Dat je dijagram konteksta. Proces 0 je dekomponovan na dva procesa (1 i 2). Na osnovu pravila balansa dekomponovati ulazne i izlazne tokove podataka procesa 0.

Pravilo balansa kaže da unija ulaznih (izlaznih) tokova u proces na jednom nivou mora biti jednaka uniji ulaznih (izlaznih) tokova u njegove potprocese na nižim nivoima. Ulazni tokovi u proces 0 su t1 i t4 dok su izlazni t2 i t3. U procese 1 i 2 ulazni su tokovi podataka t5, t6, t8 i t9 dok su izlazni t7, t10 i t11. Međutim, t7 i t8 su tokovi podataka unutar samog procesa 0 pa se oni ne mogu na ovom dijagramu dekomponovati. Dakle važe slijedeće relacije:

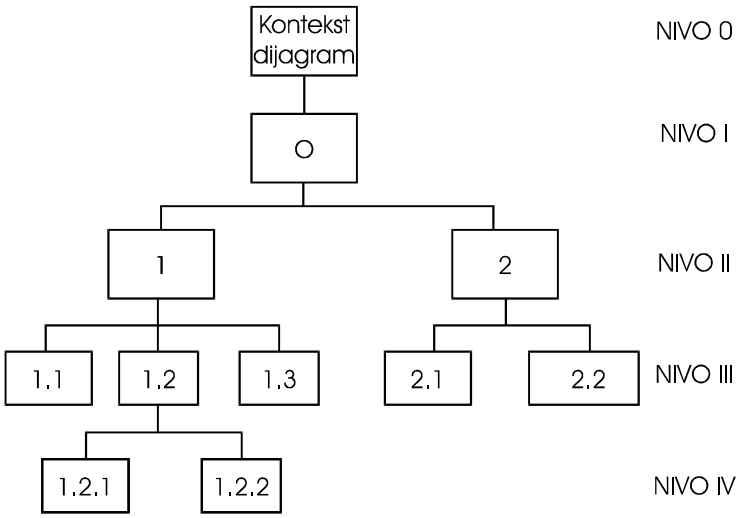
Za ulazne procese: $t1 \cup t4 = t5 \cup t6 \cup t9$

Za izlazne procese: $t2 \cup t3 = t10 \cup t11$



6. Nacrtati hijerarhijsku organizaciju procesa. Osnovni dijagram (dijagram konteksta) se može razložiti na procese 1 i 2, dok se proces 1 razlaže na procese .1, .2 i .3 dok se proces 2 razlaže na procese .1 i .2. Potproces 1.2 se može razložiti na još dva procesa. Gdje se u dokumentaciji navodi ovaj dijagram.

Ovaj dijagram je prva stranica dokumentacije u SSA analizi. U konkretnom slučaju izgleda kao na narednoj slici.



7. Dati izgled formulara riječnika podataka za tok podataka. Naziv toka je Prenos-novca, sinonimi su TP11 i Prenos. Tok podataka se može dekomponovati na tokove T1, T2 i T3 koji se uvijek javljaju i tok T4 koji je opcioni a može se ponavljati do 5 puta. U napomeni upisati Element riječnika podataka nije prekontrolisan. (SLIČNI ZADACI MOGU BITI FORMULISANI I ZA OSTALE ELEMENTE RIJEČNIKA PODATAKA).

8. Dati prikaz pseudokoda za sekvencu, selekciju i ciklus.

SISTEM:	SSA:
ANALITIČAR: **Ime studenta koji je kreirao podatak**	DATUM: ***

NAZIV TOKA
PODATAKA _____ **Prenos novca**

SINONIMI: _____ **TP11, PRENOS**

KOMPOZICIJA: _____ **T1+T2+T3+0{T4}5**

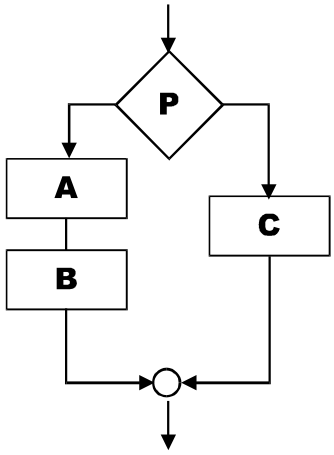
NAPOMENA: _____ **Element riječnika podataka nije prekontrolisan.**

9. Dati prikaz pseudokoda procesa koji ako je P ispunjeno izvršava operacije A pa operacije B, dok ako nije izvršava C.

U pseudokodu se ovo može zapisati kao:

IF P
A
B
ELSE
C

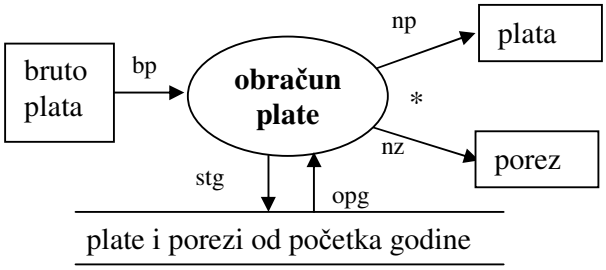
Dok je grafički prikaz oblika kao na slici:



10. Dati pseudo kod upisa studenta. Koraci u proceduri su: 1. Student se prijavljuje, 2. Služba provjerava dokumente, Ova operacija može imati četiri ishoda: a) Ako ima nedostataka student ih mora otkloniti ako ih ne otkloni odbijena mu je prijava, b) Ako student nije završio srednju školu odbijen je, c) Ako nije završio odgovarajuću srednju školu mora polagati diferencijalni, d) Ostali studenti se upućuju direktno na prijemni, 3. Studenti koji polože diferencijalni upućuju se na prijemni, 4. Studenti koji polože prijemni mogu da se upišu, 5. Razmatraju se žalbe studenata koji nijesu upisani.

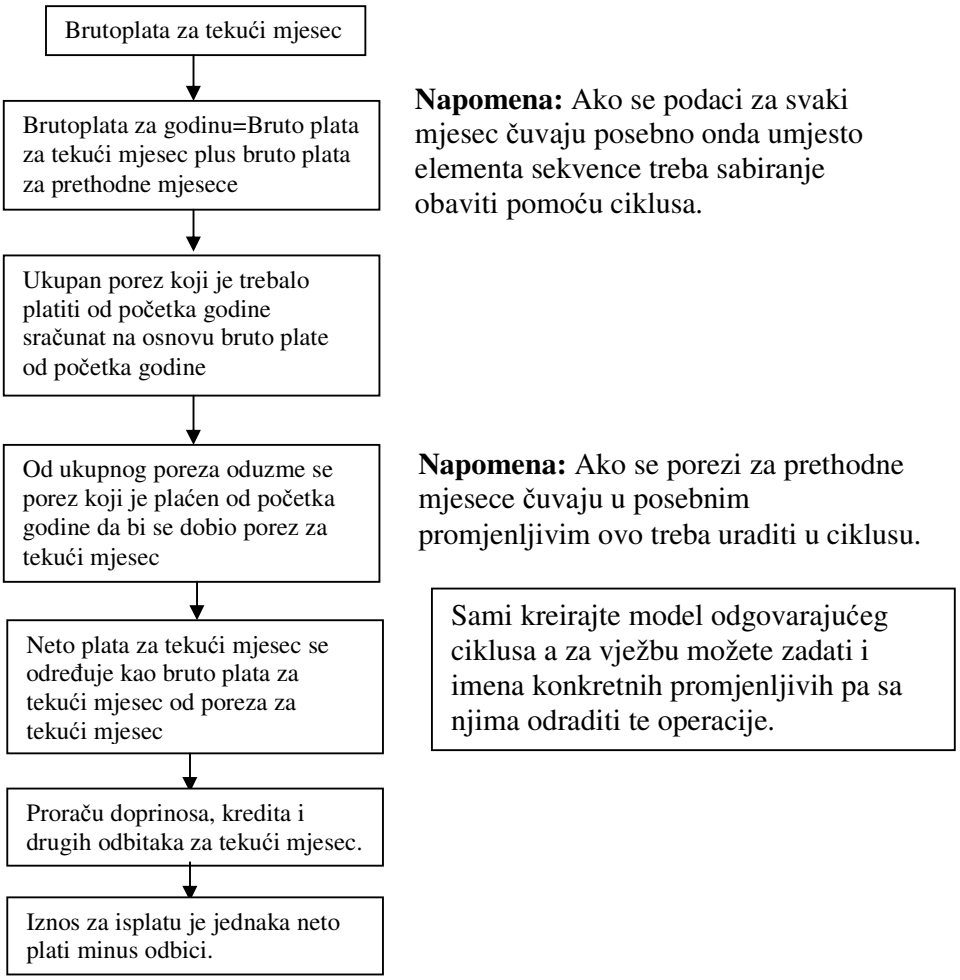
11. Dat je proces obrade neto plata koji se sastoji u sljedećem. Izvrši se proračun bruto plate. Sabere se bruto plata za tekući mjesec sa bruto platama od početka tekuće godine. Proračuna porez koji bi trebalo platiti na taj bruto iznos. Od te vrijednosti poreza oduzme ona vrijednost poreza koja je plaćena od početka godine zaključno sa prethodnim mjesecom. Tako dobijena cifra se odbije od plate kao porez za tekući mjesec. Izvrši se proračun doprinosa. Doprinosi se oduzmu od ostatka plate. Na kraju se oduzme od plate iznos kredita i drugih odbitaka i na taj način dobije neto plata.
(a) Nacrtati dijagram toka podataka za dati proces.
(b) Diskutovati organizaciju skladišta i podatka koje treba skladištiti kod datog procesa.
(c) Dati pseudo kod datog procesa.
(d) Dati element rječnika podataka koji opisuje dati proces.

(a)



(b) Očigledno je da su osnovni podaci koje treba skladištiti porezi i plate od početka godine ili barem njihovi kumulativni iznosi. Ovi podaci su potrebni za obračun plate. Stoga skladište sa podacima od početka godine mora komunicirati sa ovim procesom.

(c)



Napomena: Ako se podaci za svaki mjesec čuvaju posebno onda umjesto elementa sekvence treba sabiranje obaviti pomoću ciklusa.

Napomena: Ako se porezi za prethodne mjesece čuvaju u posebnim promjenljivim ovo treba uraditi u ciklusu.

Sami kreirajte model odgovarajućeg ciklusa a za vježbu možete zadati i imena konkretnih promjenljivih pa sa njima odraditi te operacije.

(d) Element rječnika podataka koji opisuje dati proces može da izgleda kao što je dat dolje (treba imati na umu da je formular po pravilu veličine A4)

SISTEM: Finansijsko knjigovodstvo	SSA: No.315
ANALITIČAR:	DATUM: 27.04.2005

NAZIV SKLADIŠTA: **Plate i porezi od početka godine**

NAZIV PROCES: **Obračun plate**

NUMERIČKA OZNAKA: **12**

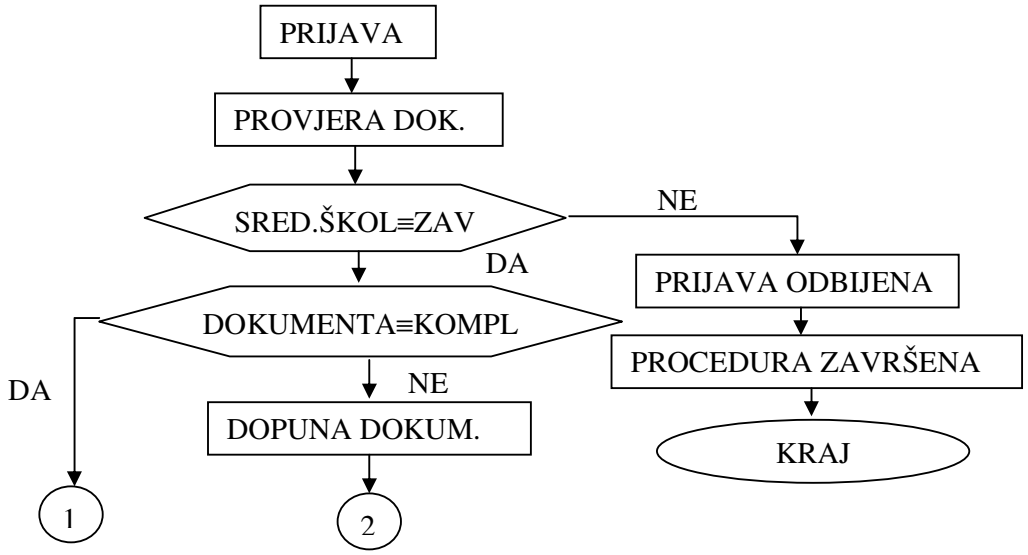
ULAZNI TOKOVI: **opg, bp**

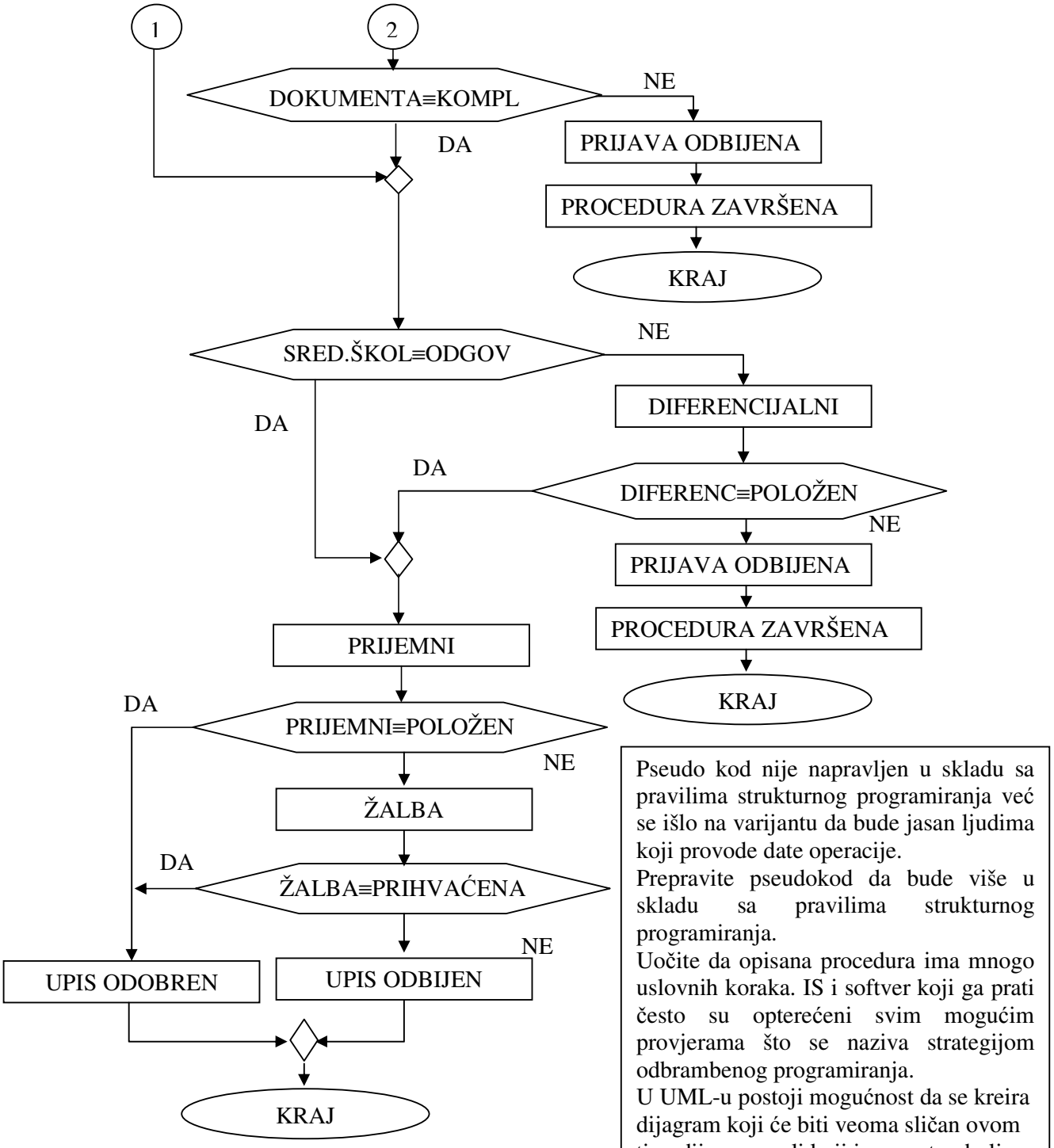
IZLAZNI TOKOVI: **stg, np, nz**

OPIS: **Opis može biti kolokvijalan u obliku teksta a često je recimo pseudo kod!!! Pošto smo mi već uradili pseudokod možete pretpostaviti da je ovdje taj pseudokod unesen. Često se koristi redundantni opis i tekstualan i kolokvijalan. Na primjer tekstualni opis bi mogao biti: Proces za obračun plate i poreza za tekući mjesec za zaposlene radnike. Proces komunicira sa skladištem Plate i porezi od početka godine. Proces dobija kao ulazni podatak bruto platu bp i na osnovu podataka iz skladišta obračunava platu i porez za tekući mjesec. Prilikom obračuna plate koristi se procedura data Zakonom o ličnim primanjima, član 67 i uredba Vlade broj 301. Izlaz iz procesa predstavlja obračunatu platu (tok np) i obračunati porez (tok nz).**

NAPOMENA: **Element rječnika podataka je test primjer. Format dokumenta nije odgovarajući.**

12. Dati pseudo kod upisa studenta. Koraci u proceduri su: **1.** Student se prijavljuje, **2.** Služba provjerava dokumente, Ova operacija može imati četiri ishoda: **a)** Ako ima nedostataka student ih mora otkloniti ako ih ne otkloni odbijena mu je prijava, **b)** Ako student nije završio srednju školu odbijen je, **c)** Ako nije završio odgovarajuću srednju školu mora polagati diferencijalni, **d)** Ostali studenti se upućuju direktno na prijemni, **3**. Studenti koji polože diferencijalni upućuju se na prijemni, **4**. Studenti koji polože prijemni mogu da se upišu, **5**. Razmatraju se žalbe studenata koji nijesu upisani.



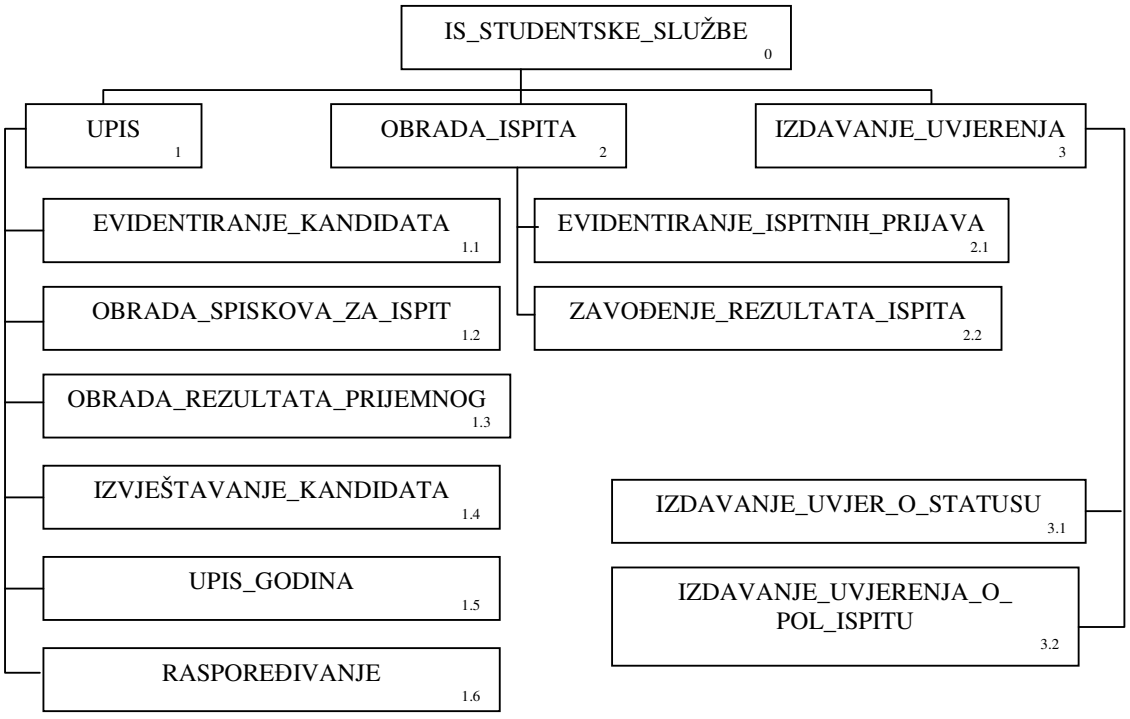


Pseudo kod nije napravljen u skladu sa pravilima strukturnog programiranja već se išlo na varijantu da bude jasan ljudima koji provode date operacije.
Prepravite pseudokod da bude više u skladu sa pravilima strukturnog programiranja.
Uočite da opisana procedura ima mnogo uslovnih koraka. IS i softver koji ga prati često su opterećeni svim mogućim provjerama što se naziva strategijom odbrambenog programiranja.
U UML-u postoji mogućnost da se kreira dijagram koji će biti veoma sličan ovom tipu dijagrama ali koji ima znatno bolje organizovana pravila od prilično nedefinisanog pseudokoda.

13. Prikazati formular rječnika podataka za skladište podataka. **SSA** šifra formulara je **KOL2**, sistem je **PROBAX**, Analitičar stavite vaše ime i prezime, kao i današnji datum. Naziv skladišta je Kartoteka studenata a sinonimi su Kartoteka i **SKL11**. Način pristupa je direktan a podaci koji čine skladište su **Ime**, **Prezime**, **Adresa**, **Broj Telefona** i **Status**. Kao napomenu postaviti da je revizija svih skladišta u sistemu planirana uskoro.

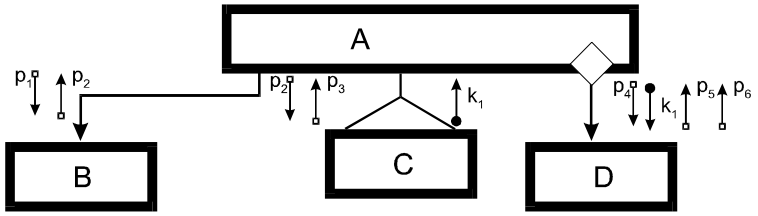
SISTEM: PROBAX	SSA: KOL2
ANALITIČAR:	DATUM: 13.05.2005
NAZIV SKLADIŠTA:	**Kartoteka studenata**
SINONIMI:	**Kartoteka, SKL11**
NAČIN PRISTUPA:	**Direktan**
KOMPOZICIJA:	**Ime+Prezime+Adresa+ Broj-Telefona+Status**
NAPOMENA:	**Revizija svih skladišta u sistemu se planira uskoro**

14. Prikazati hijerarhiju DTP-a za informacioni sistem studentske službe. Studentska služba suštinskoi posjeduje tri procesa: Upis, Obradu ispita, Izdavanje uvjerenja. Proces upisa podazumijeva operacije (procese): Evidentiranje kandidata, Obrada spiskova za ispit (podrazumijeva se postojanje prijemnog ispita koji je na mnogim fakultetima ukinut), Obrada rezultata prijemnog, Izvještavanje kandidata, Upis godine (može biti prve ali i ostalih godina) i Raspoređivanje (izrada rasporeda i eventualna podjela na grupe). Obrada ispita podrazumijeva: Evidentiranje ispitnih projava i Zavođenje rezultata ispita. Izdavanje uvjerenja poznaje dva tipa uvjerenja: Izdavanje uvjerenja o status i Izdavanje uvjerenja o položenim ispitima. U duhu reforme visokog obrazovanja došlo je do izvjesnih izmjena u ovim procesima. Što bi ukinuli, što izmjenili a što dodali u okviru navedenog ISa.

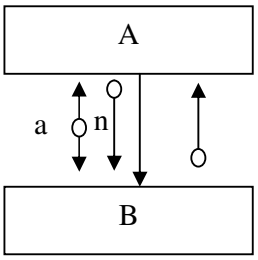


DSM

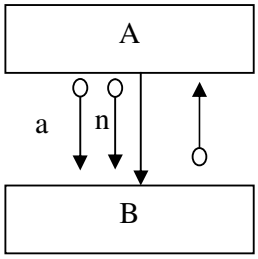
1. Formirati dijagram povezivanja modula programa. Program A, poziva modul B, pa modul C koji je ugrađen u kod i na kraju u zavisnosti od nekog logičkog uslova modul D. Prilikom pozivanja modula B proslijeđuje mu se podatak p1 dok se vraća podataka p2. Modulu C se proslijeđuje podatak p2 a vraća se podatak p3 i kontrolni signal k1. Modulu D se prosljeđuje p4 i kontrolni signal k1 a modul vraća podatke p5 i p6.



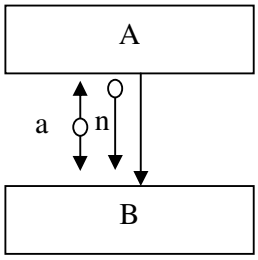
2. Posmatrajte program **A** koji je napisan u programskom jeziku C (ili C++ ili nekom srodnom). U programu je definisan vektor **a** koji je dužine **n**. Vektor i dužina vektora su argumenti funkcije **B** koja se poziva iz programa **A** i koja ima zaglavlje **int funk(int *a,int n)**. Nacrtati vezu ova dva modula i dati komentar o njihovoj povezanosti.
Ponovite isti zadatak ako je funkcija oblika **int funk(const int *a,int n)** kao i kad je funkcija **void funk(int *a,int n)**.



Funkciji se preda pokazivač na **a** (to je najvjerovatnije niz) i **n** (najvjerovatnije dužina niza). Promjene na nizu mogu uticati na glavni program pa je **a** ujedno potencijalni rezultat funkcije. Pored toga funkcija je deklarisana da vraća rezultat koji nije imenovan. Pošto promjene na nizu koji je argument funkcije mogu uticati na glavni program to možemo reći da su ovi moduli povezani zajedničkom oblašću u memoriji ili povezani strukturom podataka (nizom). Ponekad je teško praviti razliku. U svakom slučaju ovi moduli nijesu idealno razdvojeni.



Jedina ali bitna razlika u odnosu na prethodni primjer je činjenica da se u ovoj funkciji ne može vršiti promjena argumenta **a** koja bi imala uticaj na glavni program. Stoga su ovi moduli bolje razdvojeni nego oni u prethodnom slučaju pa možemo reći da su povezani podacima (mada postoje neki elementi i povezivanja strukturom podataka). U svakom slučaju riječ je o značno slabijoj vezi nego što je to bilo u prethodnom slučaju (ovdje je to vrlina a ne mana).



Ovaj primjer je gotovo u potpunosti isti kao prvi primjer osim što funkcija u ovom primjeru ne vraća rezultat (deklarisana je kao void) pa nema neimenovanog podatka od funkcije ka pozivajućem modulu. Ostali zaključci su kao u prvom primjeru.

3. Dat je sljedeći kod programa u programskom jeziku C. Dati dijagram strukture modula za ovaj program. Za dati dijagram strukture modula odrediti FAN IN i FAN OUT. Komentarisati povezanost ovih modula.

```
#include <stdio.h>
int sum(int *,int);
int max(int *,int);
void sort(int *,int);
main(){
int a[10],n,i,s;

scanf("%d",&n);
for(i=0;i<n;i++) scanf("%d",a+i);

s=sum(a,n);

for(i=0;i<n;i++) s=max(a,i);

if(n<5) sort(a,n);

`
```

A

```
int sum(int *a,int n){
int s=0, i;
for(i=0;i<n;i++) s+=a[i];
return s;
}
```

C

```
int max(int *a,int n){
int s=a[0],i;
for(i=1;i<n;i++)
if(a[i]>s) s=a[i];
return s;
}
```

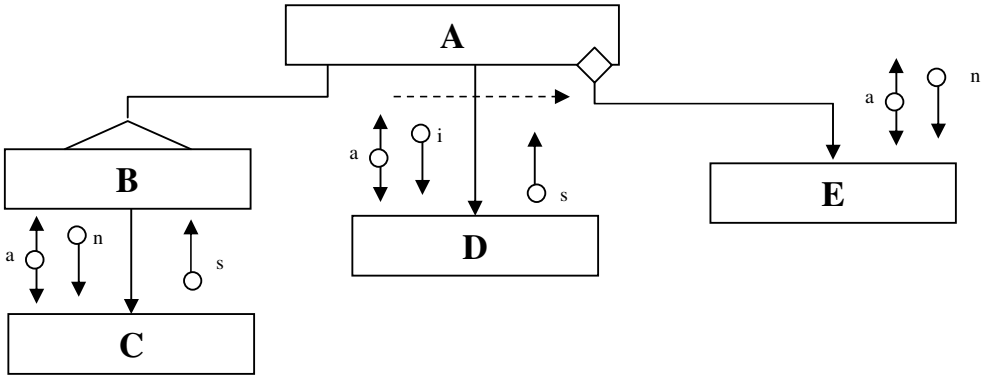
D

```
void sort(int *a,int n){
int i,j,pom;
for(i=0;i<n-1;i++)
for(j=i+1;j<n;j++)
if(a[i]>a[j]) {
pom=a[i];a[i]=a[j];
a[j]=pom;}

}
```

E

B



Modul A poziva module B, D i E a njega ne poziva nijedan modul. FAN-IN je dakle 0, dok je FAN OUT 3.

Modul B poziva modul C a njega poziva modul A. FAN-IN=FAN-OUT=1.

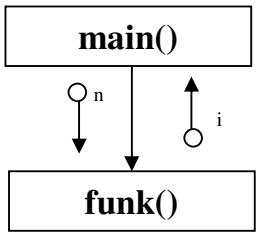
Ostale module u sistemu poziva po jedan modul dok oni ne pozivaju druge module. Dakle, FAN-IN je 1, dok je FAN-OUT 0.

Komentar povezanosti modula dajte na osnovu prethodno urađenog zadatka.

4. Dat je kod programa u programskom jeziku C. Dajte, DSM za ovaj program kao i komentar povezanosti.

#include <stdio.h>

```
int i=1;
int funk(int);
main(){
int j=2;
j=funk(j);
printf("%d %d\n",i,j);
j=funk(j);
printf("%d %d\n",i,j);
}
int funk(int n){
i+=n;
return i;
}
```



Na prvi pogled ova dva modula su povezana podacima. Međutim, podaci dijele istu globalnu promjenljivu **i** koja je deklarisana van bilo koje druge funkcije u sistemu (globalne promjenljive se deklarišu na ovaj način). Dakle, ova dva modula su povezana zajedničkom oblašću u memoriji. Ovakav način povezivanja modula se smatra lošim i treba ga izbjegavati kad god je to moguće.

5. Dati primjer programskog modula koji ima internu memoriju.

Pod internom memorijom podrazumjevamo one module kod kojih dio promjenljivih nastavlja da živi u memoriji i nakon završetka modula. Ove promjenljive se mogu koristiti po novoj aktivaciji modula. Dakle, jedan dio promjenljivih se ne ponaša na uobičajeni način (uobičajeni način podrazumjeva dealociranje promjenljivih iz memorije na kraju aktivacije modula). U programskom jeziku C ovakve promjenljive se deklarišu kao statičke. Primjer modula sa statičkim promjenljivim:

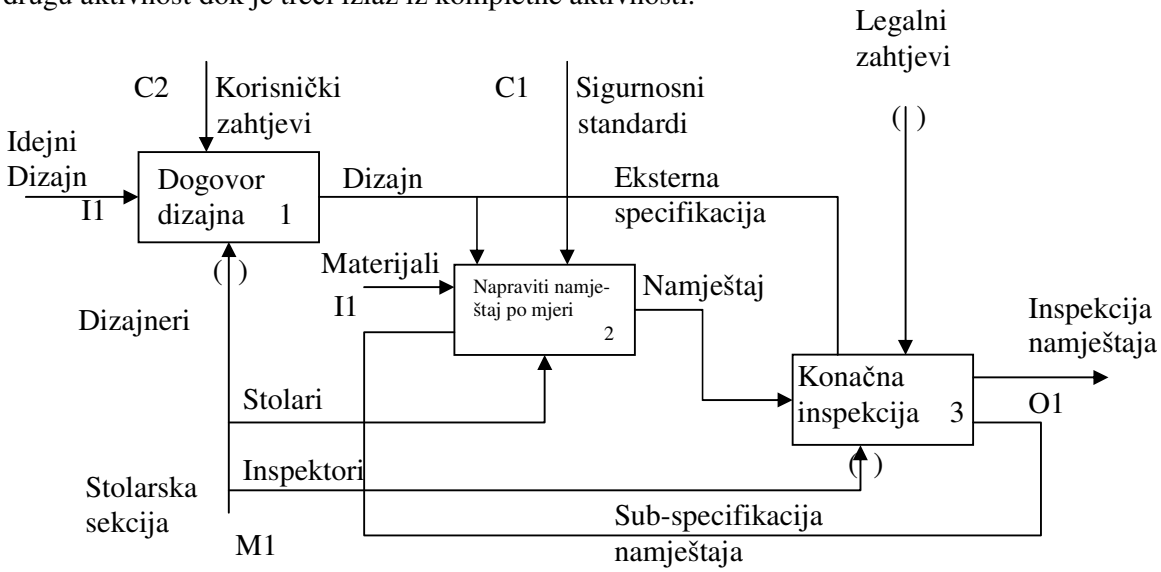
```
int funk(){
static int i=100;
i++;
return i;
}
```

Prvi poziv ove funkcije daje rezultat 101, naredni poziv daje rezultat 102 pa zatim 103 itd. Naime, statička promjenljiva nastavlja da živi i nakon kraja funkcije. U narednom pozivu funkcije preskače se deklaracija jer je ona jednom već obavljena a nema potrebe deklarisati (zauzeti memoriju) više puta jednu promjenljivu, tako da se 101 uvećava za 1 itd.

Prethodna dva zadatka opisuju situacije koje mogu da daju izuzetno elegantna rješenja ali programi mogu biti izuzetno nečitki i teški za održavanja. Kao primjer propratite treći zadatak i pokušajte da odgonetnete što će biti odštampano u programu.

IDEF0 i IDEF1X metodologije (barem pet stranica)

1. Prikazati ICOM dijagram postupka kreiranja namještaja po mjeri. Proces se sastoji od tri aktivnosti: Dogovaranja dizajna, Pravljenja namještaja i Finalne inspekcije. Ulazi u aktivnosti su redom Ideje za dizajn (prva aktivnost), Materijali i povratni rezultat iz treće aktivnosti (nazovimo ga Subspecifikacija namještaja) su ulazi za drugu aktivnost, i Namještaj kao izlaz iz druge aktivnosti je ulaz u treću. Kontrole za pojedine aktivnosti su: Zahtjevi korisnika (prva aktivnost), Specifikacija posla (jedan od izlaza prve aktivnosti) i Sigurnosni standardi (za drugu aktivnost), Eksterna specifikacija (izlaz iz prve aktivnosti) i Zakonski zahtjevi (za treću aktivnost, Zakonski zahtjevi je tunelovana aktivnost koja se ne vidi na roditeljskom dijagramu). Jedini izlaz koji do sada nije pomenut je Kontrolisani namještaj (izlaz iz treće aktivnosti). Mehanizmi na ovom dijagramu su Dizajneri (prva aktivnost), Stolari (druga aktivnost) i Inspektori (treća aktivnost). Sva ova tri mehanizma potiču iz mehanizma Stolarska sekcija. Prvi i treći mehanizam su tunelovani. Izlazi iz pojedinih aktivnosti su Dizajn (izlaz iz prve aktivnosti) koji se dijeli na Radnu specifikaciju i Eksterna specifikacija (redom kontrole druge i treće aktivnost), Namještaj (izlaz iz druge aktivnosti), Podspecifikacija namještaja i Provjereni namještaj (prvi od ovih namještfaja je ulaz za drugu aktivnost dok je treći izlaz iz kompletne aktivnosti.



2. Pretpostavimo da u informacionom sistemu imamo dvije aktivnosti A1 vezanu za održavanje podataka o nekim važnim elementima sistema i aktivnost A2 vezanu za očuvanje podataka o šifrarnicima. U okviru aktivnosti A1 koriste entiteti E1 (CRUD entitet sa atributima a1 – IUN, a2 IRUN, a3 UN), E2 (CRU entitet sa atributima a4 – IRUN, a5 IRN, i a6 IU). U okviru aktivnosti A2 postoje entiteri se E3 (CRUD entitet sa atributima a7 IUN, a8 IU, a9 N, a10, RU), i E4 (CRU entitet sa atributima a11 – IRU, a12, RU i a13 R). Prikazati CRUD-IRUN matricu za ovaj dio ISa.

Naziv aktivnosti	Naziv entiteta	CRUD	Naziv atributa	IRUN
A1	E1	CRUD	a1	I UN
			a2	IRUN
			a3	UN
	E2	CRU	a4	IRUN
			a5	IR N
			a6	I U
A2	E3	CRUD	a7	I UN
			a8	I U
			a9	N
			a10	RU
	E4	CRU	a11	IRU
			a12	RU
			a13	R

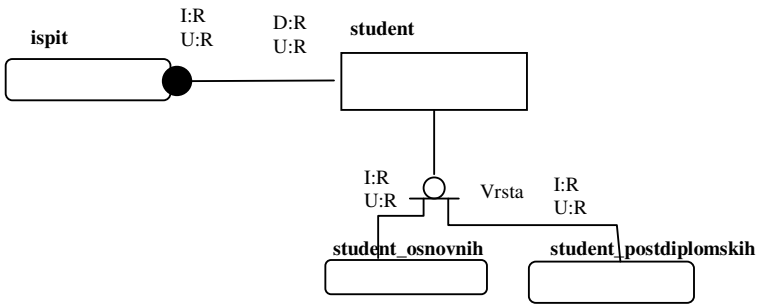
3. Objasnite prikaze sljedećih validacionih pravila za poslovne domene u Microsoft Accessu ili nekom sličnom programu (uz olakšicu da nije potrebno prikazivati validaciona pravila vezana za eventualne Null vrijednosti za sljedeće domene: malo slovo, veliko slovo, cifra, slova i znaci blanko, alfanumerički karakteri, 8 karaktera, četvorocifreni broj, pozitivni broj, ne više od 100%, današnji ili prethodni datumi, polje u koje se mora nešto upisati, moguće vrijednosti 'M' ili 'Ž', moguće vrijednosti 1, 2, 4, 8, polje koje može biti Da, Ne ili Null, E-mail adresa.

Rješenja:

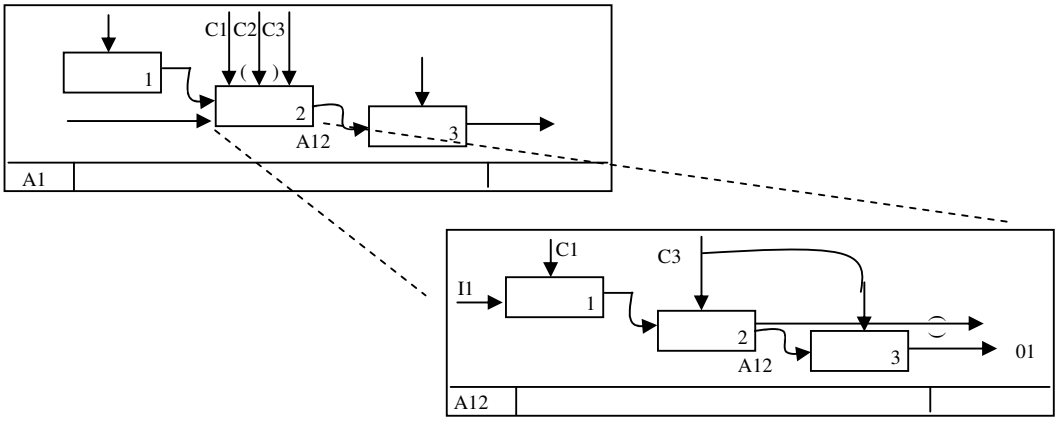
Mala slova	Between 'a' And 'z'
Velika slova	Between 'A' And 'Z'
Cifra	Between '0' And '9'
Slova i znaci blanko	[Between 'a' And 'z'] OR [Between 'A' And 'Z'] OR ' '
Alfanumerički karakteri	[Between 'a' And 'z'] OR [Between 'A' And 'Z'] OR [Between '0' And '9']
8 karaktera	????????
4-cifreni broj	Between 1000 And 9999
Pozitivan broj	>0.0
Ne više od 100%	>-1.0 And <1.0
Današnji i prethodni datumi	<=Date()
Neprazno polje	Not Null
Moguće vrijednosti 'M' i 'Ž'	IN ['M', 'Ž']
Moguće vrijednosti 1, 2, 4, 8	IN [1, 2, 4, 8]
Da, Ne ili Null	IN ["Da", "Ne"] OR Null
E-mail adresa	*@*.*

Ovo su moguče simplifikovane verzije validacionih pravila. U softverskim alatima ova pravila su često preciznija ali zato komplikovanija. Npr. pravilo za e-mail adresu u Microsoft Access-u je Is Null OR ((Like "*?@?*.?*") AND (Not Like "*[,;]*")).

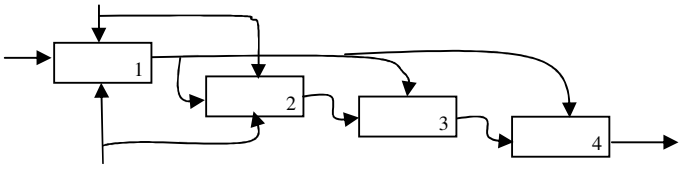
4. Klasa Student ima sljedeće podređene entitete: student_osnovnih_studija i student_ postdiplomskih studija. Ove dvije klase su vrste studenta "is a". Prema ova dva entiteta ostvaruju se veze koje su U – Cascade, I – Restrict i D – Cascade. Student "polaže" Ispit-e. Veza studenta sa klasom Ispit je U – Restrict, I – Restrict i D – Cascade. Prikazati vezu entiteta student sa ostalim entitetima u sistemu:



5. Kreirajte IDEF0 dijagram aktivnosti A1 u kojem se nalaze aktivnosti 1, 2 i 3. Aktivnost 1 ima jednu kontrolu nema ulaza i ima jedan izlaz. Ulazi u aktivnost 2 su izlaz iz aktivnosti 1 i ulaz u sistem. Aktivnost 2 ima 3 kontrole C1, C2 i C3 od kojih je C2 tunelovana. Izlaz iz aktivnosti 2 je ulaz u aktivnost 3. Aktivnost 3 ima jednu kontrolu a izlaz iz ove aktivnosti je izlaz iz kompletnog sistema. Dekomponovati aktivnost A12 na tri podaktivnosti. Prva podaktivnost ima ulaz I1 i kontrolu C1. Druga aktivnost prima ulaz iz prve aktivnosti ima kontrolu C3 i produje dva izlaza od kojih je jedan izlaz tunelovan van aktivnost A12 dok je drugi ulaz u aktivnost 3. Aktivnost 3 ima kontrolu C3 i izlaz O1.



6. Što je prikazano na sljedećem dijagramu opisati kolokvijalno.



Rješenje. Aktivnost je dekomponovana na 4 aktivnosti. Ulaz u sistem je ujedno ulaz u aktivnost 1 koja ima jednu kontrolu koja je zajednička za kontrolu 2 i jedan mehanizam koji je ponovo zajednički za kontrolu 2. Izlaz iz prve aktivnosti je ulaz u kontrolu 2 a ujedno je kontrola za aktivnosti 3 i 4. Ostale aktivnosti čine ravan tok. Izlaz iz aktivnosti 2 je ulaz u aktivnost 3, izlaz iz aktivnost 3 je ulaz u aktivnost 4 dok je izlaz iz aktivnosti četiri izlaz iz kompletne aktivnosti.

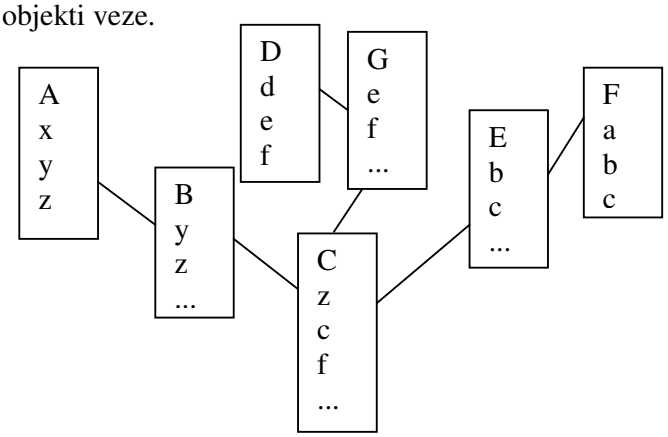
7. Obradite u sklopu seminarskog rada priču o referencijalnom integritetu. Možete da sumirate značaj integriteta kroz sljedeće teze:
(a) Popravka kvaliteta podataka;
(b) Brži razvoj;

(c) Manje grešaka;
(d) Očuvanje konzistentnosti kroz aplikacije.

Kao početni materijal u trenutku pisanja ovog materijala može da posluži web-dokumentacija:
http://www.odbms.org/download/007.02%20Blaha%20Referential%20Integrity%20Is%20Important%20For%20Databases%20November%202005.PDF

Napominjem da postoje izuzetno dobri web-materijali i projekat vezan za on-line dokumentaciju o bazama podataka i informacionim sistemima koje treba obavezno koristiti u svom razvoju kao programera, stručnjaka za baze podataka i informacione sisteme.

8. Posmatrajmo primjer sljedećeg dijagrama objekti veze.



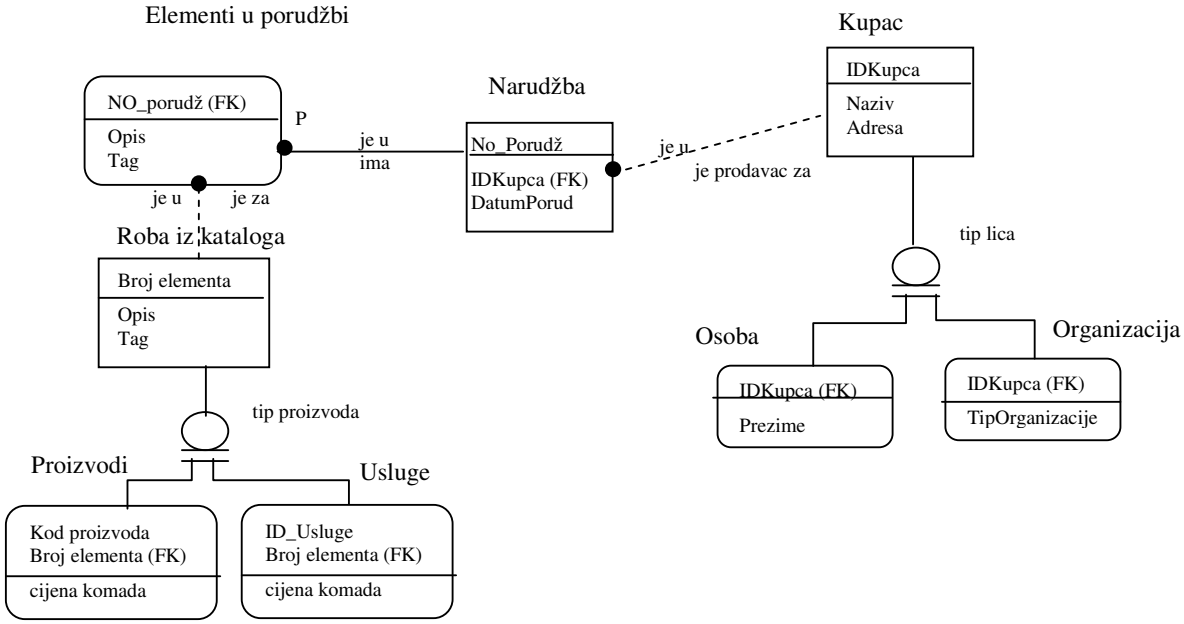
Veza objekta A prema podređenim klasama je Delete Cascade, veza F prema E je No Action dok je prema klasi C Cascade ili No Action. Posljednja varijanta je situacija kada je veza između D i G kaskadna ali je veza G prema C No Action. Što se dešava u sistemu ako dođe do brisanja elemenata u tabelama A, D i F.

Rješenje. Brisanje elementa iz tabele A dovodi do kaskadne povezanosti u tabelama B i C odnosno podaci o tom elementu se brišu u ovim tabelama (tabela B ostaje bez podatka y i z dok tabela C ostaje bez podatka z). U slučaju da se zaključa tabela A dolazi do zaključavanja korespondentnih podataka u tabelama B i C jer se podaci mogu promijeniti iz bilo koje od ovih tabela. Prilikom brisanja F-a ne mogu se brisati oni redovi u tabeli koji su referencirani u E dok oni koji nisu referencirani mogu. Pošto se redovi iz E ne mogu izbrisati to se ne mogu izbrisati ni redovi iz C tabele. Kada se primjeni zaključavanje tabele F dolazi do zaključevanja ove tabele za upis, dok se tabela E zaključava za čitanje a nema restrikcija koje su primjenjene nad tabelom C. U slučaju brisanja tabele D situacija je sljedeća. Premda je veza D-G kaskadna činjenica da je druga veza u ovom nizu ograničena čini i ovu vezu restrikovanom. Ako je red u tabeli D referenciran u G taj se red u D ne briše. Zapamtite da veza između G i C koja je No action ne utiče ni na koji način na ostale veze prema C iz drugih tabela. Kada se zaključavanje primjeni na A on je zaključan za operaciju pisanja dok je jedina tabela koja joj je direknto podređena (u ovom slučaju tabela B) zaključana za čitanje. Tabela C u ovom slučaju nije zaključana. Više detalja vezanih za zaključavanje, uspostavljanje veza, i ograničenja pri vezama možete naći (u trenutku pisanja ovog materijala) na odličnom IBM-ovom sajtu:

http://publib.boulder.ibm.com/infocenter/rbhelp/v6r3/index.jsp?topic=%2Fcom.ibm.redbrick.doc6.3%2Fwag%2Fwag29.htm

9. IDEF1X metodologija je i dan danas jedna od najupotrebljavanijih metodologija za modelovanje koncepata u okviru IS-a. Danas primat polako prelazi u ruke UML-a ali zapamtite da je IDEF1X veoma dobro prilagođenja relacionim modelima (i veoma slabo strategijama koje izlaze van ovog

okvira) koji su izuzetno važni. Jedan od najvažnijih korisnika i promotera IDEF1X je vlada SAD i njene vladine službe pa to dobrim dijelom garantuje da će ova strategija zadržati svoj značaj i u godinama koje slijede. Ilustrujmo ovu strategiju kroz jedan model. Model treba da prikaže vezu entiteta Kupac (može biti Osoba ili Organizacija a smata se da se ne može pojaviti neki drugi entitet van ove dvije grupe) koji vrši porudžbu preko entiteta Faktura Robu. Sva Roba se nalazi u Katalogu koji čine Stvari i Usluge. Ponovo nije moguće napraviti podjelu koja bi uključila neku drugu vrstu roba. Kreirati Entity-Relationship dijagram u skladu sa IDEF1X metodologijom. Povesti računa o tipovima veza, kardinalnosti i itd.

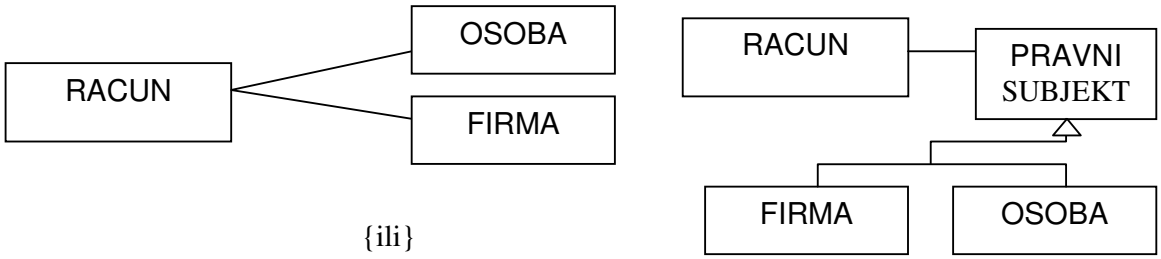


UML

1. Izvršiti grafički prikaz klase: Automobil koja ima sljedeće atribute: tipAutomobila koji je tipa Marka, godinaStarosti tipa Godina, pređenaKilometraža tipa Km. Nad automobilom treba implementirati operacije koje odgovaraju na pitanje o predviđenim atributima kao i postavljanje atributa. Takođe, treba predvidjeti operaciju procjenaVrijednosti koja na osnovu predviđenih datih atributa treba da procjeni vrijednost automobila. Izlazni argumenat ove operacije treba da bude tipa Novac.

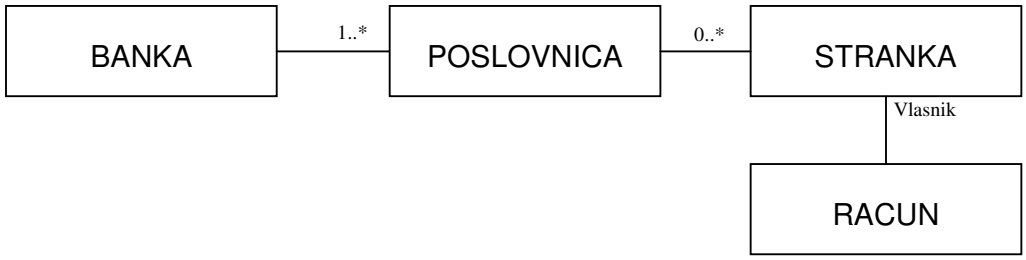
Automobil
tipAutomobila: marka godinaStarosti: godina pređenaKilometraža: km
<<podaci >> starost(): godina tip(): marka kilometraža():km <<postavljanje>> postaviTip(t: marka) postaviStarost(s: godina) postaviKm(k: km) <<vrijednost>> procjena(): novac

2. a) U informacionom sistemu banke klasa Račun može da bude otvorena od dvije vrste subjekata: fizičkih lica (klasa Osoba) i kompanija (klasa Firma). Prikazati direktnu vezu klase Račun sa ovim dvijema klasama i predvidjeti odgovarajuće višestrukosti.
b) U praksi je način povezivanja demonstriran u prethodnom primjeru često veoma nepogodan jer se uvijek kada je to moguće izbjegava veza jednog objekta sa više sličnih objekata. Naime, ako bi se desila promjena u načinu komunikacije sa jednim objektom to bi imalo uticaj na softversku implementaciju svih djelova. Stoga se često uvodi nova klasa i preko nje ostvaruje komunikacija sa stvarnim klasama. Klasa koja je uvedena (naziva se osnovnom) može sama po sebi da ima značenje (da se mogu kreirati objekti te klase) ali i ne mora. Ako nema značenje to se naziva apstraktna klasa. Klase koje su povezane sa osnovnom klasom nazivaju se izvedenim i ta relacija je zapravo generalizacija. Prikazati ovakvu vezu za slučaj modela koji je prethodno opisan.

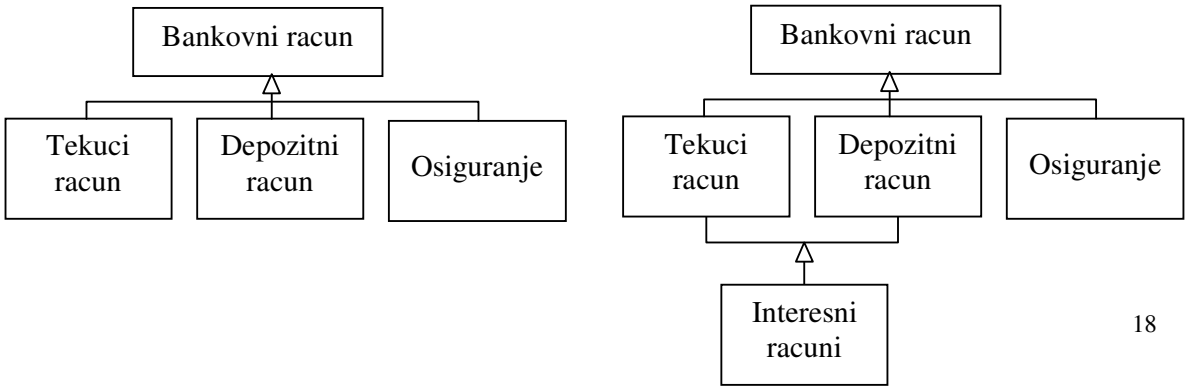


3. Prikazati UML model veze između BANKE i RAČUN. Banka može imati jednu ili više poslovnica, svaka poslovnica ima nijednu ili više stranaka dok svaka stranka ima tačno jedan račun. Stranka je vlasnik računa.

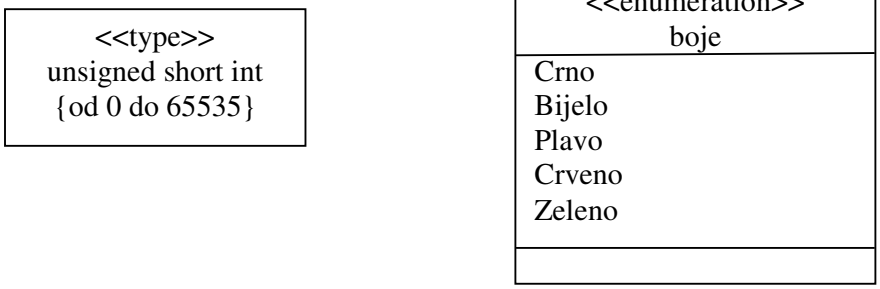
Očigledno nam trebaju četiri klase (entiteta): BANKA; POSLOVNICA, STRANKA i RAČUN. U vezi između STRANKA i RAČUN uloga klase Stranka je vlasnik. To se prikazuje sljedećim modelom (sve veze koje su upotrijebljene su asocijacije).



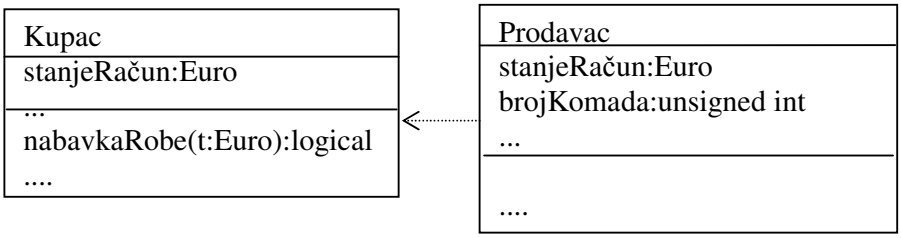
4. a) Modelovati sljedeću generalizaciju: Bankarski računi se mogu podijeliti u tri kategorije: Tekući računi, Depozitni računi, Osiguranja.
b) Modelovati sljedeću generalizaciju: Bankarski računi se mogu podijeliti u sljedeće kategorije: Tekući računi, Depozitni računi, Osiguranja. Interesni računi mogu biti podvrsta Tekućih i Depozitnih računa.



5. Modelovati primitivni tip podatka unsigned short int koji uzima vrijednosti od 0 do 65535. takođe modelovati enumeraciju (nabrajanje) boje koja može uzimati vrijednosti: Crno, Bijelo, Plavo, Crveno, Zeleno.



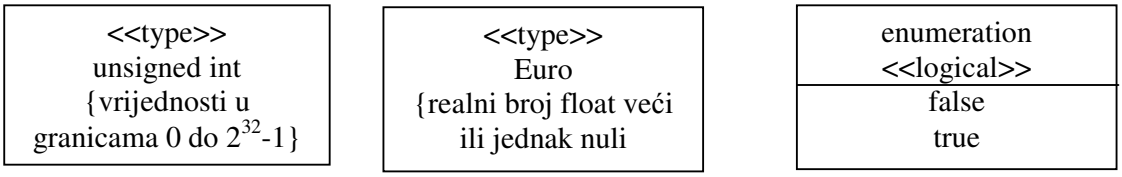
6. Uvedite klasu kupac. Kupac ima više atributa od kojih je najvažniji Stanje na računu. Kupac ostvaruje trgovinu od klase Prodavac. Klasa Prodavac ima više atributa među kojima je Stanje na računu i Broj komada. Klasa kupac posjeduje operaciju: NabavkaRobe koja ima atribut cijenu robe u Eurima. Ovom operacijom se stanje na računu kupca smanjuje za dati iznos u Eurima odnosno ako Kupac nema toliko stanje na računu funkcija se ne obavlja (vraća rezultat false a u slučaju uspješnog ishoda true). Napisati pseudo kod sličan bilo kom programskom jeziku za implementaciju predmetne funkcije. Promjenljive false i true su implementirane kao stereotip (nabrajanje) logical, tip Euro je realni broj veći ili jednak nuli zaokružen na dvije decimale dok se broj komada označava u cijelim brojevima većim od nule.



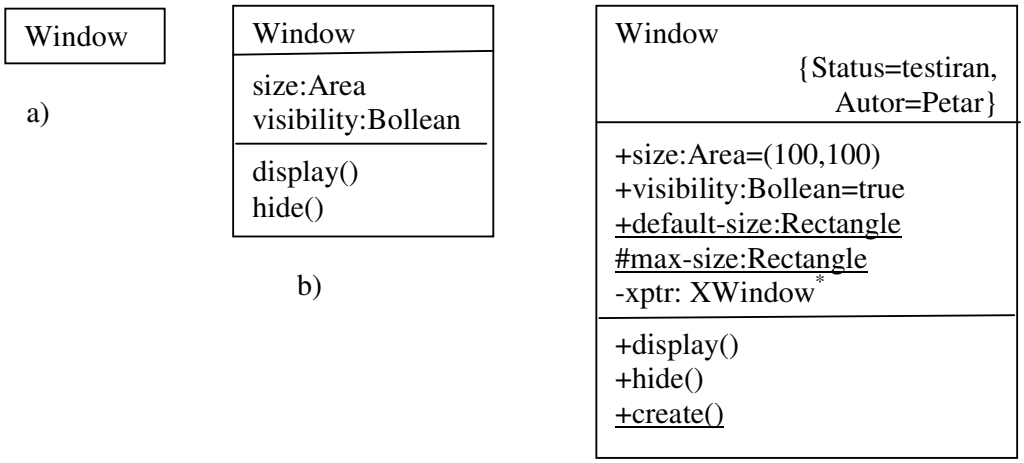
Na stanje klase Kupac očigledno utiče stanje u klasi Prodavac naravno ako Kupac obavi datu porudžbu stanje na računu u klasi prodavac se mijenja kao i broj komada na skladištu. Stoga smo vezu modelovali kao zavisnost. Ovdje se može reći da postoji ciklična veza između i ove dvije klase. Ovo se ne praktikuje u UML-u već sve ciklične veze ukazuju da se može uvesti nova klasa npr. Narudžba preko koje klase Kupac i Prodavac mogu da komuniciraju. Psuedo kod funkcije nabavkaRobe(t:Euro): logical bi mogao da bude:

```
Kupac::nabavkaRobe(t:Euro):logical
if(t>stanjeRacun) return false;   //kupovina nije obavljena
else
stanjeRačun= stanjeRačun-t;            //umanjujemo stanje na računu
...        //dodatne operacije koje u vlasništvo kupca uvode dati predmet
...        //dodatne operacije koje stanje na računu prodavca povećavaju i smanjuju broj komada npr.
Prodavac::stanjeRačun= Prodavac::stanjeRačun+t;
Prodavac::brojKomada= Prodavac::brojKomada-1;
//moguće je da ni prodavac nije imao robu na skladištu u praksi i to treba provjeriti.
return true;
```

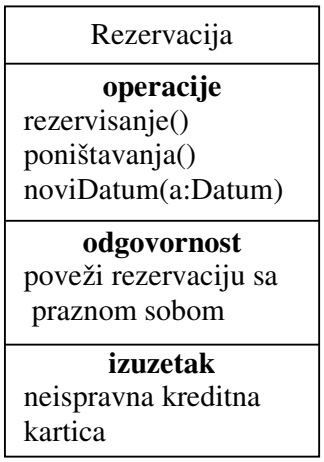
endif
end function
Implementacija odgovarajućih stereotipa može da bude:



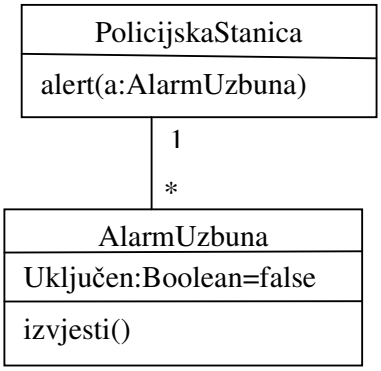
7. Izvršiti prikazivanje klase Window na tri nivoa apstrakcije: a) Prikazati samo klasu Window bez detalja; b) Klasa Window ima osobine size koje je tipa Array i visibility koje je tipa Boolean kao i metode display i hide. c) Klasa Window je testirana a autor joj je Petar, podatak size je javni sa predefinisanom vrijednošću (100,100), podatak visibility je zaštićen sa predefinisanom vrijednošću true, za čitavu klasu se predviđa javni podataka default-size koje je tipa Rectangle kao i zaštićeni podatak maximum-size istog tipa, pored ovoga klasa posjeduje i privatnu promjenljuvu xptr koje je tipa pokazivač na XWindow. Funkcije display i hide su javne. Za čitavu klasu postoji zajednička javna funkcija create. Ove tri funkcije ne primaju argumente. Pored njih postoji i privatna funkcija attachXWindow koja prima argument tipa XWindow.



8. Modelovati klasu Rezervacija. Klasa posjeduje operacije: rezervisanje, poništavanje i noviDatum. Jedino operacija noviDatum ima argument koji je tipa Datum. Odgovornost ove klase je da poveže rezervaciju sa dostupnom sobom. Klasa ima i jedan izuzetak kada se rezervisanje ne može obaviti a to je slučaj neispravnog broja kreditne kartice. Dati prikaz ove klase sa nazivima pojedinih blokova.



9. Modelovati vezu između klasa PolicijskaStanica i AlarmUzbuna. Policijska stanica ima jednu operaciju alert koja prima argument tipa AlarmUzbuna. Jedna policijska stanica je povezana sa više alarma. Veza između ove dvije klase je asocijativna. Klasa AlarmUzbuna ima promjenljivu Uključen koja je tipa Boolean i postavljena je na false i operaciju izvijesti. Dati i pseudokod operacije uzbunjivanja stanice.



Pseudokod operacije izvijesti bi mogao biti:

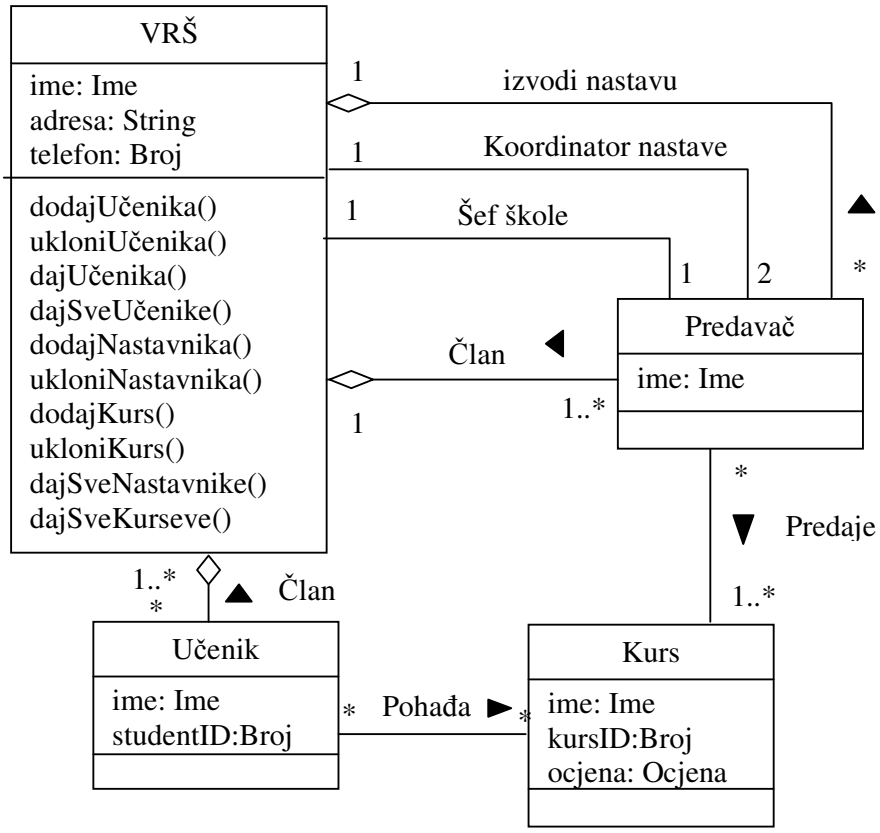
```
if Uključen==true
station.alert(this)
```

gdje this u smislu jezika C++ znači ovaj AlarmUzbuna odnosno signalizira stanici koji je alarm izazvao uzbunu.

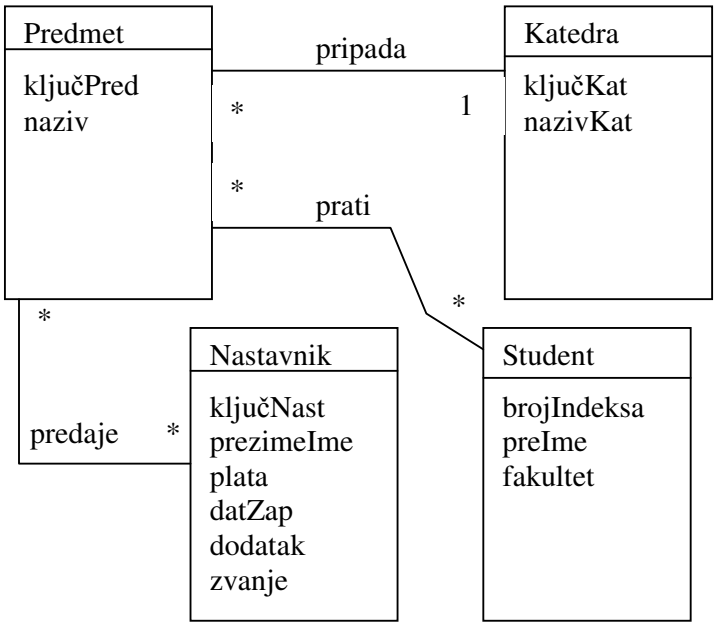
10. Prikazati model VRŠ. VRŠ ima klase Student, Nastavnik (jedan od nastavnika je šef škole), Službu, Računovodstvo. Student može uvesti sve neophodne dodatne Klase (npr. Predmet, Asistent, Koordinator itd). Izvršiti modelovanje na različitim nivoima apstrakcije.

Na slici je data gruba verzija modela. Za samostalni rad studenti treba da urade sljedeće:

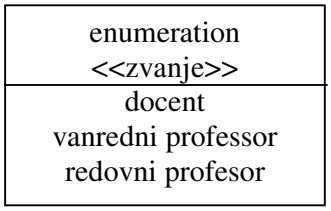
a) Kreirati stereotip za tip Ocjena.
b) Dodati tip Studentska služba. Neka Studentska služba bude povezana sa Studentima preko klasa prijava ispita i ovjera semestra. Neka studentska služba bude povezana sa nastavnikom preko klase ispitna prijava.
c) Izmjenite prethodnu realizaciju i neka Studenska služba bude povezana i sa studentom i sa predavačem preko klase Kurs. Ako je potrebno proširiti klasu Kurs da obuhvati Ispitnu prijavu i ovjeru semestra.
d) Koja vam se od prethodne dvije realizacije čini praktičnijom i lakšom za proširenje.
e) Treba za svaki kurs implementirati promjenljivu koja predstavlja nastavni plan i program kursa. Izvršite to i kreirajte odgovarajući stereotip.
f) Dodati klasu Računovodstvo koja prima uplate studenata. Proširiti ostale klase da omoguće rad sa ovom klasom.
g) Umjesto implementacije klase Računovodstvo da li se može implementacijom ograničenja u nekoj od postojećih klasa izvršiti kontrolu studentskih uplata i da se npr. onemogući polaganje ispita studentu koji nije izmirio obaveze prema školi.
h) Koje od prethodnih rješenja vam se čini praktičnijim.
i) Implementirati promjenljive koje su zajedničke za pojedine klasa a koje sadrže broj ukupan broj nastavnika, kurseva i studenata.



11. Kreirati klase podataka Predmet, Nastavnik, Katedra i Student. Klasa Predmet je povezana sa klasom Katedra preko veze imenovane sa "pripada" sa klasom Nastavnik preko veze "predaje" a sa klasom Student preko veze položio. Klasa Student ima osobine brojIndeksa, preIme i fakultet. Klasa Nastavnik ima osobine ključNast, prezimeIme, plata, datZap, dodatak, zvanje. Predmet ima osobine ključPred i naziv dok klasa Katedra ima osobine ključKat i nazivKat. Dati dijagram klasa sa specificiranim osobinama. Na dijagramu treba odrediti tip relacije između pojedinih klasa kao i višestrukost pojedinih veza. Zatim za svaku od klasa odrediti odgovarajuće metode i uvesti tipove podataka za pojedine osobine koje opisuju argumente metoda kao i osobine pojedinih klasa. Tipova podataka treba da bude što manje. Uvesti stereotipe za sve tipove podataka za koje je to moguće. Dati pseudokod realizacije svih metoda koji su uključeni u model.



Analizirajmo tipove podataka koji se mogu koristiti za pojedine podatke članove: ključevi (ključPred, ključKat, ključNast) bi trebali da budu cijeli brojevi jer operacije koje provode nad cijelim brojevima su brže od operacija nad ostalim tipovima podataka. brojIndeksa premda vrlo vjerovatno isto ima namjenu da bude korišćen kao ključ u nekoj operacije je po prirodi niz karaktera jer ima kosu crtu u sebi. Nazivi (naziv i nazivKat), prezimeIme, preIme, fakultet su kao stvoreni da budu nizovi karaktera. Podaci kao što su plata i dodatak mogu biti realni brojevi zaokruženi na dvije decimale ako se odnose na platu u novčanom iznosu (često je u pitanju koeficijenat ili neki sličan identifikator koji je opet najčešće realni broj). Finalno datZap je kandidat da bude tipa Datum, Date ili nešto slično. Osobina zvanje može biti predstavljena nabrajanjem koje se realizuje putem odgovarajućeg enumeratora. Na primjer nastavnička zvanja kod nas su "docent", "vanredni profesor", "redovni profesor" premda postoji mogućnost da se zvanja klasifikuju i na neki drugi način (na primjer ponekad postoji zvanje "predavač").



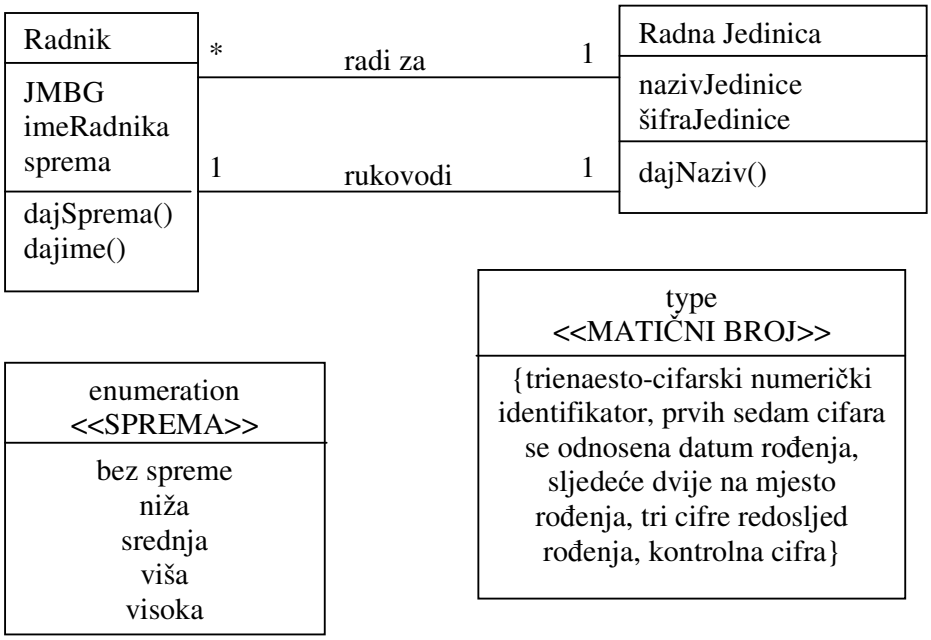
U ovako postavljenom sistemu koji je veoma rudimentaran tako da u njemu ne postoje podaci članovi namjenjeni stvarnoj obradi već su gotovo svi ovi članovi samo namjenjeni metodima kao što su inspektori i mutatori: postavi vrijednost odnosno pročitaj vrijednost.

Za vježbu ažurirajte klasu Nastavnik tako da sadrži podatak o datumu posljednjeg izbora zvanje. Funkcija unaprijedi() treba da za nastavnike kojima je isteklo pet godina od posljednjeg izbora izvrši unapređenje odnosno da ih iz zvanja "docent" unaprijedi u zvanje "vanredni profesor" i iz zvanja "vanredni profesor" u zvanje "redovni profesor". Pretpostaviti da imate funkciju Date() koja vraća tekući datum i da imate funkciju ProtokVremena() koja prima dva argumenta koji su datumi i vraća broj godina koji je prošao između dva datuma. Prilikom ažuriranja zvanja treba promjeniti datum posljednje izbora. Funkcija Date() je nečlanicaa dok je funkcija ProtokVremena() privatna funkcija članica.,

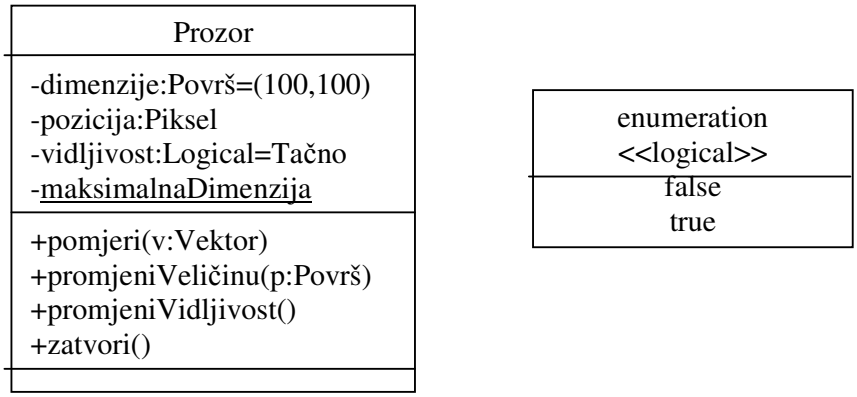
Dakle u klasi Nastavnik dakle treba dodati funkciju unaprijedi() koja je javna i ProtokVremena(Datum, Datum):int koja je privatna i koja prima dva argumenta koji su datumi i koja vraća cijeli broj koliko je godina prošlo između dva datuma. Datum posljednjeg izbora u nastavnićkog zvanje može biti promjenljiva datIzbora tipa Datum. Funkcija unaprijedi() onda može da ima sljedeću realizaciju:

```
Nastavnik::unaprijedi()
if ProtokVremena(datIzbora,Datum())>=5
if zvanje=="docent"
zvanje="vanredni profesor"
datIzbora=Datum()
else if zvanje=="vanredni profesor"
zvanje="redovni profesor"
datIzbora=Datum()
endif
endif
```

12. Prikazati vezu klasa Radnik i Radna Jedinica. Klasa Radnik ima osobine JMBG (jedinstveni matični broj građana) imeRadnika i sprema. Klasa Radna Jedinica ima osobine nazivJedinice i šifraJedinice. Postoje dvije veze između ovih klasa. Jedna se imenuje kao "radi za" a druga "rukovodi". Prikazati dijagram klasa na datom nivou apstrakcije i odrediti odgovarajuće višestrukosti. Zatim za svaku od klasa predvidjeti odgovarajuće operacije. Nakon toga predvidjeti pojedinim argumentima operacija i osobinama klase tipove i predefinisane vrijednosti ako je to moguće. Za sve tipove za koje je to moguće uraditi predvidjeti odgovarajuće stereotipe. Za predviđene operacije dati pseudokod ili opisati njihovo izvršavanje. Implementirati za klasu Radnik osobinu koja važi za čitavu klasu i koja predstavlja ukupan broj radnika zaposlenih u datoj radoj jedinici. Povezati je sa prethodnim elementima.



13. Prikazati grafički klasu Prozor (u UML notaciji) koja ima osobine (atribute) dimenzija koja po defaultu uzima vrijednost (100,100) i tipa je Površ, pozicija koja nema default vrijednost a tipa je Piksel, vidljivost koja je tipa Logical i na početku postavljena na Tačno, maksimalnaDimenzija koja je zajednički elemenat za sve objekte date klase. Sve osobine su privatni elementi klase. Klasa ima operacije (metode) pomjeri() koja uzima argument tipa Vektor, promjeniVeličinu koja uzima argument tipa Površ, funkciju promjeniVidljivost() koja mijenja vidljivost prozora i funkciju zatvori(). Svi metodi su javni. Klasa ima odgovornost da izvrši prikaz prozora u Windows okruženju. Modelujte jedan od upotrijebljenih tipova podataka stereotipom. Kreirajte pseudokod neke od funkcija ovog modela.

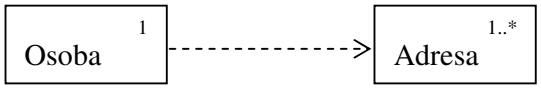


Gruba realizacija pseudokoda pojedinih funkcija bi mogla biti:
Prozor::pomjeri(v:Vektor)
pozicija=pozicija+v
endfunction

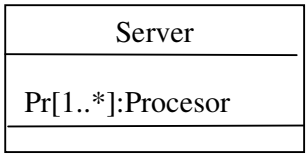
Prozor::promjeniVeličinu(p:Površ=
dimenzije=p
endfunction

Prozor::promjeniVidljivost()
if(vidljivost==Tačno)
vidljivost=Netačno
else
vidljivost=Tačno
endif
endfunction

14. Modelujte vezu između klase Osoba i klase Adresa. Svaka osoba ima jednu ili više adresa:



15. Klasom treba modelovati Server koji može da ima jednu ili više promjenljivih klase Procesor.

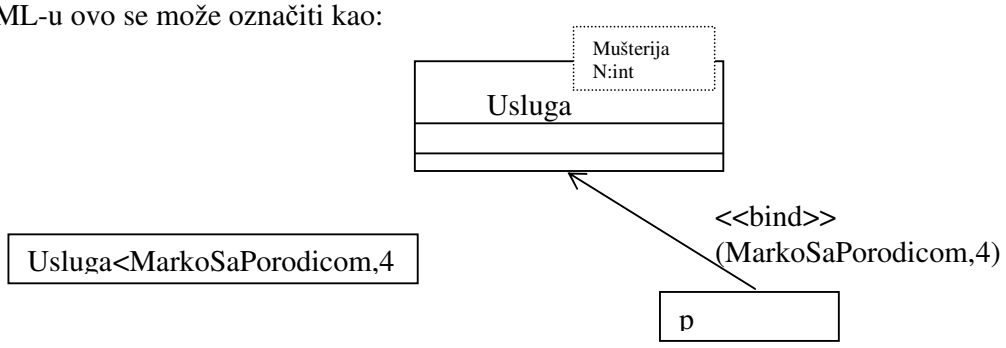


16. Uvesti klasu Usluga koja kao parametre ima tip podatka Mušterija i cijeli broj N.

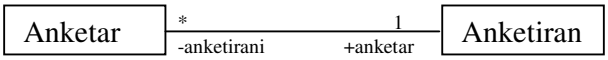
U programskom jeziku C++ ovo se može zapisati kao:

```
template <class Mušterija, int N>
class Usluga{
// Unutar definicije klase se moraju nalaziti metodi i funkcije koje koriste tip Mušterija i broj N
};
```

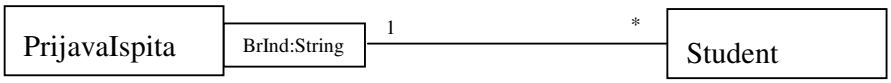
Promjenljva p koja je kreirana za ovaj šablon sa parametrima MarkoSaPorodicom i brojem 4 može se označiti kao:
p:Usluga<MarkoSaPorodicom,4>;
U UML-u ovo se može označiti kao:



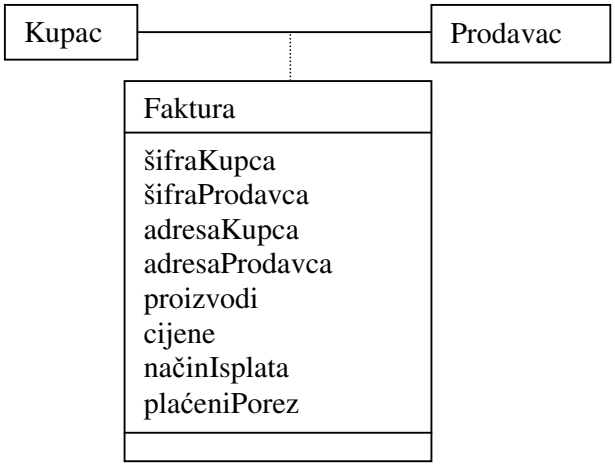
17. Modelujte vezu anketara i grupe anketiranih u sitaciji kada anketar nema vidljivost anketiranih.



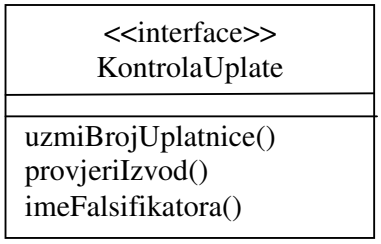
18. Ilustrovati vezu prijave ispita sa studentom gdje je student vidljiv preko kvalifikatora BrInd.



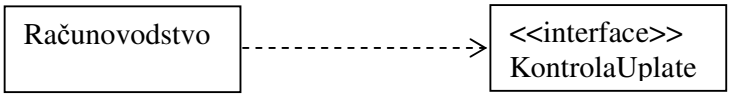
19. Ilustrujte vezu Faktura između kupca i prodavca preko klase asocijacija.



20. Dati ilustraciju interfejsa KontrolaUplate gdje se provjerava da li je predmetna uplata izvršena pravilno.



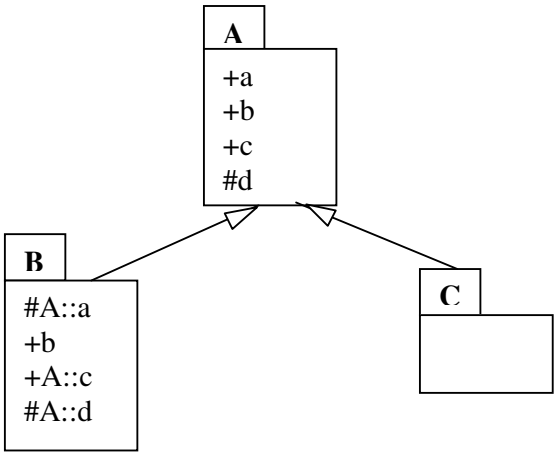
Za vježbu možete da implementirate ovaj interfejs u nekom dijagramu klasa gdje Računovodstvo vrši provjeru preko datog interfejsa.



Zavisnost je upotrebljena jer računovodstvo zavisi od toga što i kako radi navedeni interfejst kojim se provjeravaju uplate.

21. Klasa Osoba ima standardne osobine i ostvaruje vezu sa klasom Putnička agencija. Klasa Osoba može imati ulogu Radnik, Mušterija, Šef. Izvršiti odgovarajuće modelovanje pomoću interfejsa.

22. Paket A ima elemente a, b i c koji su svi javni i element d koji je zaštićen. Paket B nasljeđuje paket B dodaje novi element e a predefiniše element b iz osnovne klase (i b i e su javni). Elementu a se mijenja vidljivost i on postaje zaštićen. Paket C vrši nasljeđivanje paketa A bez promjena.

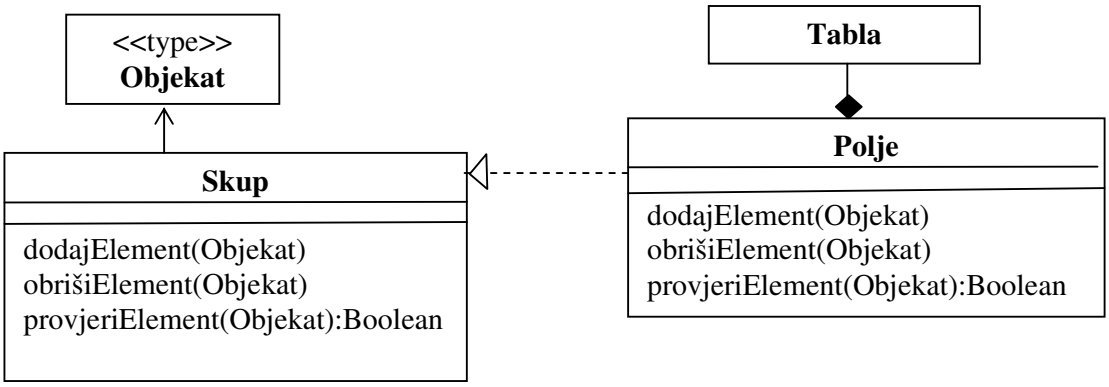


23. Za vježbu osmislite dva paketa i zatim izvršite preuzimanje elemenata jednog paketa u drugi paket.

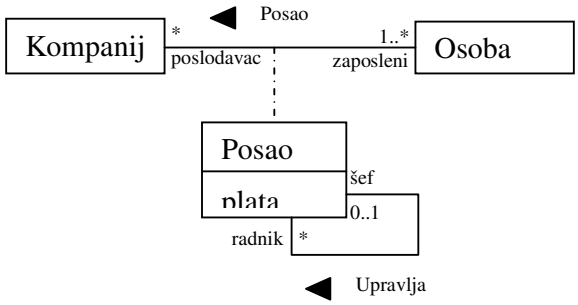
24. Kreirati klasu rezervacija koja ima sljedeće funkcije: garantiranje(), poništavanje(), promjena(). Odgovornosti klase su da ne smije da prikaže račun i da spoji rezervaciju sa slobodnom sobom. Izuzetak koji je implementiran je Neispravna kreditna kartica. Funkcija promjena uzima argument koji je tipa Datum.

Rezervacija
garantiranje() poništavanje() promjena(D:Datum)
Odgovornosti Ne pokazuj račun Poveži sa slobodnim sobama
Izuzetak Neispravna kreditna kartica

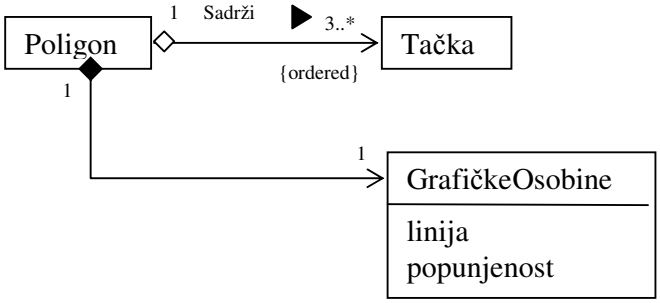
25. Kreirati sljedeći klasni dijagram koji uključuje i realizacije pojedinih klasa. Klasa Tabla sastoji se od više Polja. Klasa Polje ima sljedeće funkcije: dodajElement(), obrišiElement(), provjeriElement(), postaviVeličinu(). Prve tri funkcije uzimaju argument koji je tipa Objekat dok treća funkcije uzima argument tipa Integer. Treća funkcija vraća rezultat koji je tipa Boolean. Klasa Polje se realizuje na osnovu klase Skup. Skup ima tri funkcije koje se po svemu poklapaju sa onim kod klase Polje. Klasa Skup zavisi od klase (tipa) Objekat.



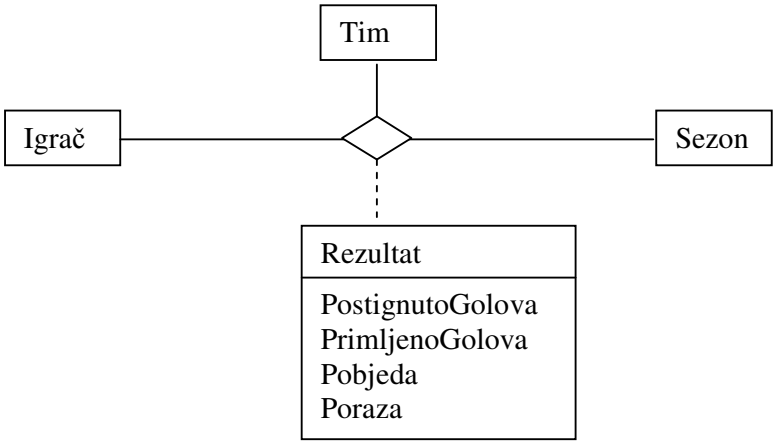
26. Modelovati vezu između Kompanije i Osobe u kojoj se kompanija pojavljuje kao poslodavac dok se osoba pojavljuje kao zaposleni. Vezu možete modelovati kao posebnu klasu Posao čiji je glavna osobina plata. Postoji mogućnost da zaposleni se pojavljuje na poslu kao Šef ili kao Radnik.



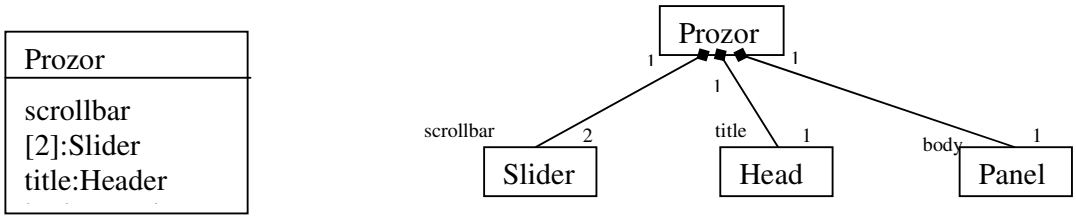
27. U CorelDraw-u ili sličnim alatima jedno od osnovnih sredstava kojim se prikazuju objekti je poligon. Poligon se sastoji od najmanje tri Tačke koje su uređene po odgovarajućem redosljedu (ovo se naglašava korišćenjem vizuelne modifikacije {ordered} u blizini klase Tačka). Takođe poligon posjeduje GrafičkeOsobine što je specijalna klasa kojom se modeluju osobine Linija i Popunjenost. Izvršiti modelovanje ovakvog objekta u UML-u.



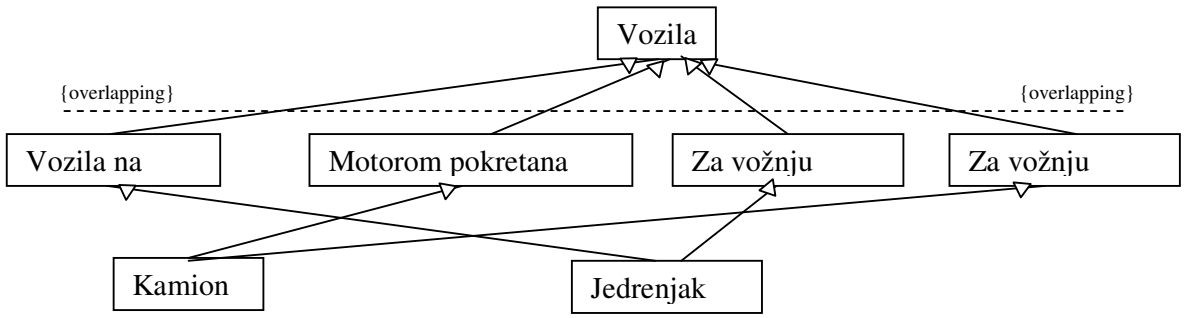
28. Prikazati ternarnu asocijaciju u vezi: Igrač, Klub, Sezona koja se može modelovati kao klasa Rezultat koja ima sljedeće podatke: PostignutoGolova, PrimljenoGolova, Pobjeda, Poraza i Nerješeno.



29. Klasa prozor se sastoji od dva skrolbara koji su klase Slider, naslovneLinije koja je tipa Header i radnePovrši koja je tipa Panel. Prikazati ovo preko standardnog klasnog dijagrama i preko agregacija.

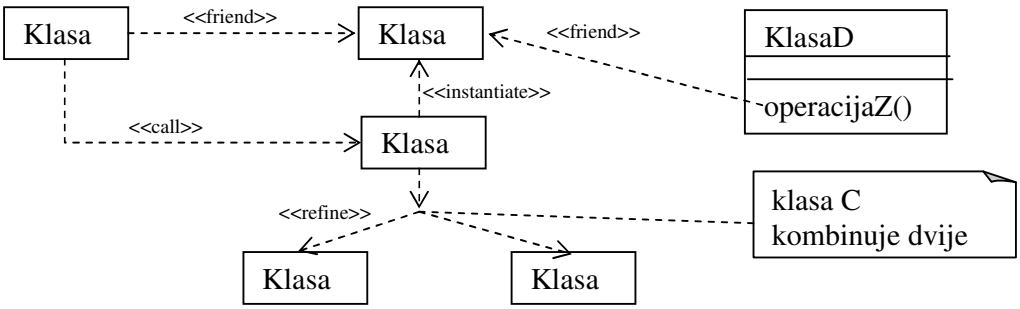


30. Demonstirati upotrebu generalizacija vozila polazeći od Jedrenjak i Kamion. Generalizacije obaviti preko načina pokretanja i puta po kojem se kreću ova vozila.

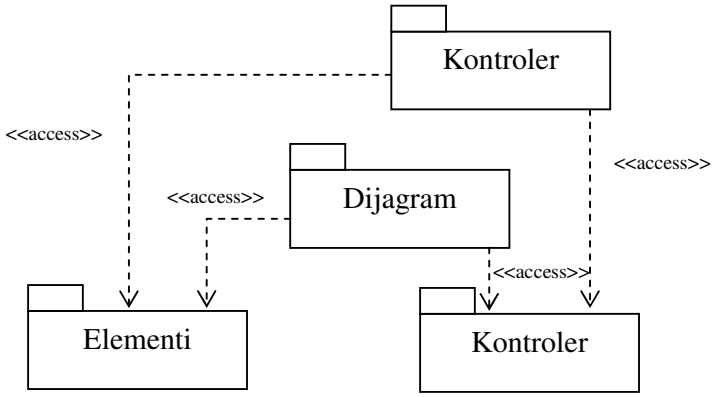


Napomena: Stereotip {overlapping} označava da se isti objekat može pojavljivati u više klasa u datoj generalizaciji. Npr. Jedrenjak je i vozilo na vjetar i namjenjen ja za vožnju vodom.

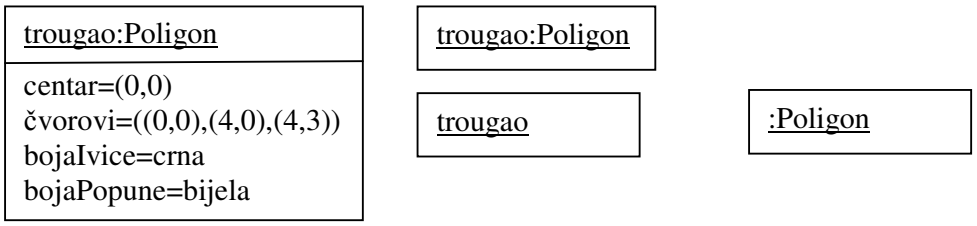
31. Na narednom primjeru razradićemo modelovanje zavisnosti u UML-u. Osnovni dio model se sastoji od klasa: KlasaA, KlasaB, KlasaC. KlasaA zavisi od klase B i klase C s time što je klasa B prijatelj klase A dok klasa A poziva klasu C. Klasa B je samo jedna instanca klase C. Klasa D zavisi od klase B, odnosno preciznije rečeno metod operationZ() član klase D je prijatelj klase B. Klasa C predstavlja kombinaciju dvije logičke klase (Klase D i Klase). Vezu klase C sa ove dvije klase treba modelovati preko stereotipa <<refine>> i preko napomena povezane preko relacije zavisnosti sa ovom vezom.



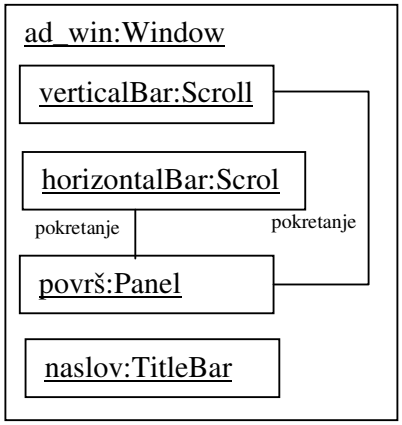
32. Prikazati vezu između paketa Kontroler, Diagram, Elementi, Grafika. Paket Kontroler ima pristup javni sadržajima svih ostalih paketa dok paket Dijagram ima pristup javnim sadržajima paketa Elementi i Grafika. Pristup javnom sadržaju se modeluje preko stereotipa <<access>>.



33. Prikazati objekat Trougao koji pripada klasi Poligon. Atributi klase poligon su: centar, čvorovi, bojaIvice i bojaPopune. Neka je centar trougla (0,0) a neka su čvorovi (0,0), (4,0) i (4,3), boja ivice neka je crna dok boja popune neka bude bijela. Prikazati ovaj objekat na različitim nivoima apstrakcije.

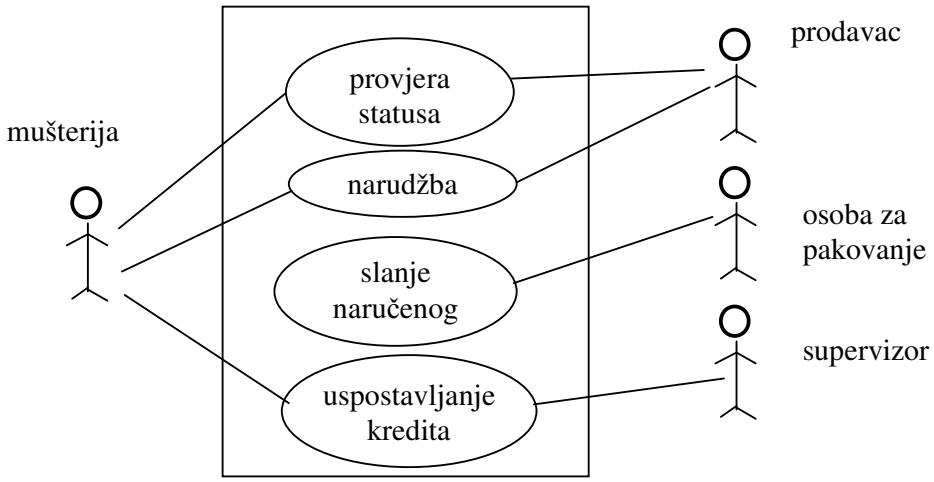


34. Objekat se može sastojati od drugih objekata. Ako se takva veza između objekat koji čine složeni objekat može vizuelizovati dijagramom objekata riječ je o kompozitnom objektu. Posmatrajmo objekat ad_win koji pripada klasi Window. Objekat ima sljedeće atribute horizontalBar i verticalScrolBar koji pripadaju klasi ScrollBar površ koja pripada klasi Panel i naslov koji pripada klasi TitleBar. Između ScrollBar-ova i panela postoji relacija zavisnosti koja se naziva pokretanje (kretanje scrollbar-a pomjera panel a moguće je i obrnuto.

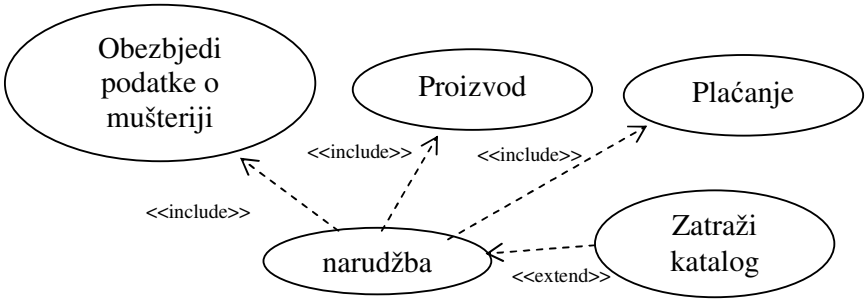


35. Na stranici 34 je dat jedan relativno složen UML model koji je kreirao Kobryn 2000 godine. Protumačite dijagram (djelove i veze između njih).

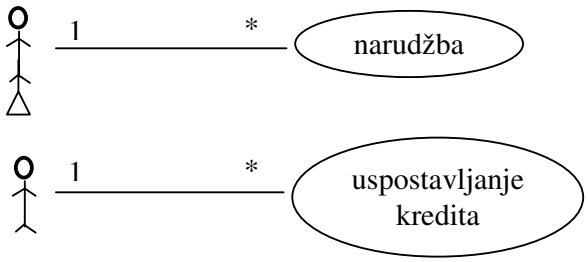
36. Prikazati dijagram korisničkih funkcija u kojem se pojavljuju sljedeće korisničke funkcije: provjera statusa, narudžbina, slanje naručenog i uspostavljanje kredita. U operacijama koje uključuju ove korisničke funkcije u obliku crne kutije učestvuje izvođač mušterija sa jedne strane i izvođači prodavac, osoba za pakovanje i supervizor sa druge strane. Mušterija vidi prvu, drugu i četvrtu korisničku funkciju dok prodavac vidi prvu i drugu. Ostala dva izvođača vide redom treću i četvrtu korisničku funkciju.

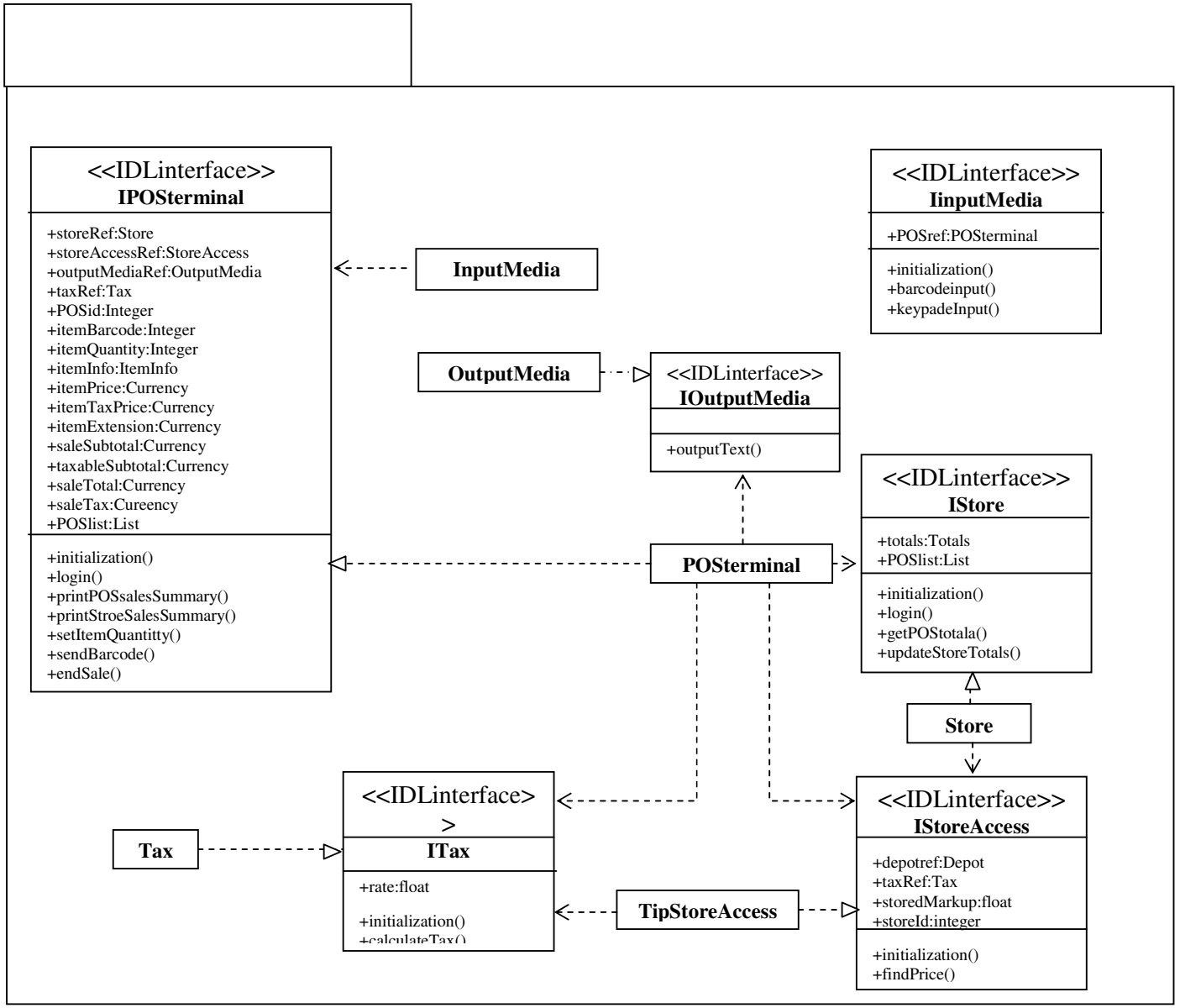


37. Prikazati detaljnije korisničku funkciju narudžba iz prethodnog primjer. Funkcija uključuje u sebi funkcije: Obezbjedi podatke o mušteriji, Proizvod, Plaćanje. Funkcija se može proširiti sa funkcijom Zatraži katalog.



38. Prikazati drugi detalj sa dijagrama korisničkih funkcija datog u zadatku 4. Supervizor je samo specijalni slučaj prodavca (njegova generalizacija). Supervizor može da rukuje sa više "uspostavljanja kredita" dok prodavac može da rukuje sa više "narudžbi".

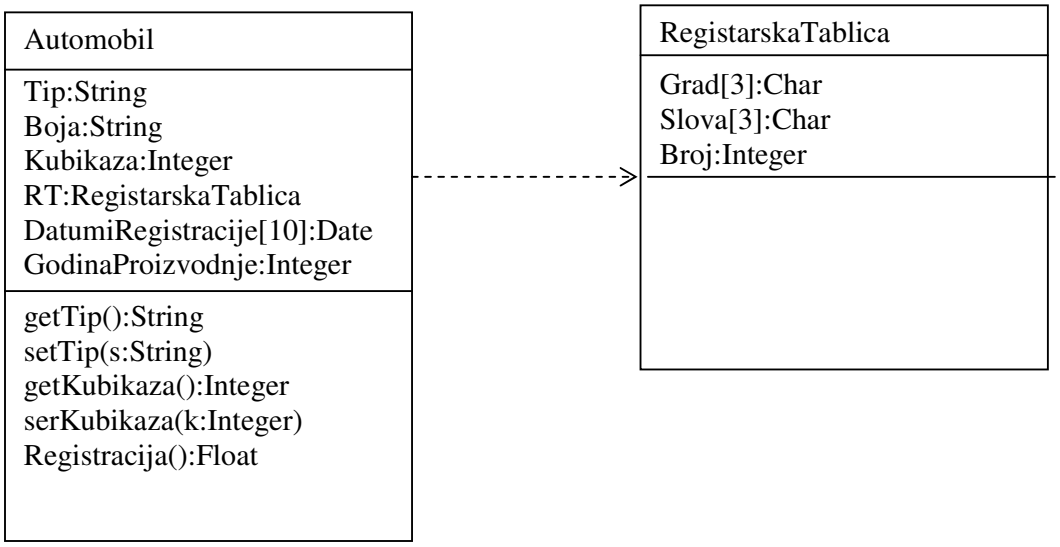



<<IDLinterface>>
IPOSterminal
+storeRef:Store
+storeAccessRef:StoreAccess
+outputMediaRef:OutputMedia
+taxRef:Tax
+POSid:Integer
+itemBarcode:Integer
+itemQuantity:Integer
+itemInfo:ItemInfo
+itemPrice:Currency
+itemTaxPrice:Currency
+itemExtension:Currency
+saleSubtotal:Currency
+taxableSubtotal:Currency
+saleTotal:Currency
+saleTax:Cureency
+POSlist:List
+initialization()
+login()
+printPOSsalesSummary()
+printStroeSalesSummary()
+setItemQuantitty()
+sendBarcode()
+endSale()
InputMedia
<<IDLinterface>>
IinputMedia
+POSref:POSterminal
+initialization()
+barcodeinput()
+keypadeInput()
OutputMedia
<<IDLinterface>>
IOutputMedia
+outputText()
<<IDLinterface>>
IStore
+totals:Totals
+POSlist:List
+initialization()
+login()
+getPOStotala()
+updateStoreTotals()
POSterminal
Store
<<IDLinterface>>
ITax
+rate:float
+initialization()
+calculateTax()
Tax
TipStoreAccess
<<IDLinterface>>
IStoreAccess
+depotref:Depot
+taxRef:Tax
+storedMarkup:float
+storeId:integer
+initialization()
+findPrice()

Zadaci sa kolokvijuma.

1. Kreirati klasu Automobil koja pored ostalih ima podatke o tipu, boji, kubikaži, registarskoj tablici, datumima registracije (može ih biti više), godini proizvodnje. Kreirati za klasu po 2 gettersa i settersa. Kreirati za klasu koja predstavlja registarsku tablicu i koja ima dva stringa od po dva karaktera i jedan trocifreni cijeli broj. Prikazati i obrazložiti vezu koja postoji između klase koja predstavlja apstrakciju automobila i klase koja predstavlja registarsku tablicu. Napisati funkciju koja računa vrijednost registracije za automobil i pravilno je pozicionirati u odgovarajuću klasu. Funkcija računa vrijednost registracije kao 10% kubikaže umanjeno za 3% po godini starosti. Ukupno umanjenje po osnovu godina starosti ne može da pređe 30% (10 godina). Na ovu vrijednost treba dodati fiksnu taksu od 50E. Recimo ako je kubikaža 1600, a godine starosti 5 vrijednost koja je potrebna da se plati za registraciju vozila je 160x85%+50=186E.

Rješenje.



Za vezu između registarske tablice i automobila upotrebljena je relacija zavisnosti najviše iz činjenice da tip podataka RegistarskaTablica pojavljuje kao tip podatak osobine klase Automobil dakle klasa Automobil zavisi od promjena u RegistarskojTablici. Nije nemoguće imati i neki drugi tip veze između ove dvije klase (automobil sadrži registarsku tablicu pa se onda može koristiti neki oblik asocijativne – agregirane relacije). U svakom slučaju odluka između asocijacije i zavnosti kojoj ovdje dajem prednost mora biti dobro osmišljena i tumačena (u uslovima kolokvijuma sa barem rečenicom dvije). Realizacija po dva odabrana gettersa i settersa je data dolje:

```
Automobil::getTip():String
return Tip

Automobil::getKubikaza():Integer
return Kubikaza

Automobil::setTip(s:String)
Tip=s

Autobil::setKubikaza(k:Integer)
Kubikaza=k
```

Vrijedi napomenuti da ako imamo tip String koji je enkapsuliran u obliku klase ili na drugi način možemo raditi putem operatora dodjele kako je urađeno gore te možemo vraćati string kao rezultat funkcije kao što je urađeno gore. Međutim, ako se string realizuje kao niz karaktera predmetne realizacije moraju biti nešto drugačije i složenije. Funkciju Registracija() smo smjestili u klasu Automobil jer se odnosi na Automobil. Ona nema

argumenata jer sve potrebne podatke dobija iz samog automobila a vraća vrijednost koju je potrebno uplatiti po ovoj našoj formuli.

```
Automobil::Registracija():Float
Pomocna:Float
Vrijeme:Integer
Pomocna=0.1*Kubikaza
Vrijeme=AktuelnaGodina()-GodinaProizvodnje
if Vrijeme<10
return Pomocna*(1-Vrijeme*0.03)+50
else
return Pomovna*0.7+50
endif
```

U okviru ove funkcije postoji specifičnost da smo morali na neki način da odredimo aktuelnu godinu. Za to smo se poslužili funkcijom AktuelnaGodina() koja može biti i funkcija nečlanica a koja vraća tekuću godinu. Napominjem da postoje slične funkcije a nije je problem ni napisati ako na sistemu postoji bilo kakva funkcija koja vraća tekući datum i ako znamo format toga datum.

Napomena. Za prethodni zadatak sa automobilom moguće je napraviti mnoštvo varijanti.

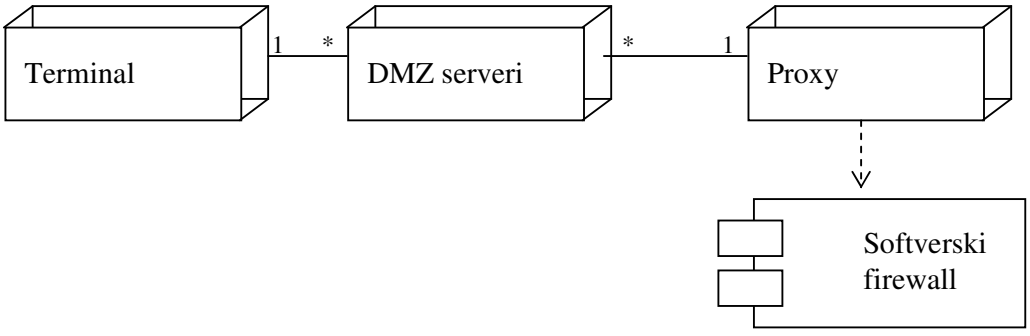
Jedna moguća varijanta je:

Kreirati klasu AUTOMOBIL koja se sastoji od podataka: BOJA, TIP, KUBIKAŽA, GODIŠTE, BROJ_ŠASIJE, BROJ_MOTORA, REGISTRACIJA (niz Datuma na koje je pravljena registracija), BROJ_REGISTRACIJA. Pojedinim podacima pridružiti odgovarajuće tipove. Barem dva tipa podataka realizujte putem odgovarajuće stereotipa. Realizovati po dva gettersa i setters. Napisati funkciju koja određuje datum kada automobilu ističe registracija. To je datum od jedne godine više nakon termina posljednje registracije. Napisati model klase DATUM koja bi se mogla koristiti za određivanje datuma. Prikazati vezu klasa DATUM i AUTOMOBIL.

2. (a) Prikazati dijagram raspoređenosti na kojem je prikazana veza: Terminal, Proxy (proxy u sebi sadrži komponentu softverski firewall) i tzv. Demilitarizovane zone (DMZ) koja je zapravo banka od nekoliko servera.
(b) Komponenta Provjera.DSP koristi dva interfejsa A.XLL, BazaVirusa.DLL. Iz drugog interfejsa koriste se funkcije provjera() i pretražibazu(). Komponenta realizuje interfejs B.DLL i interfejs SIGMA_ELIMINATOR.DLL koji ima funkcije f1(), ključ() i f2().
(c) Prikazati klasu Student i njenu vezu sa klasom Ispit. Jedan student ima više od 5 Ispita. Prikazati pravilno višestrukost i determinisati uloge ovih klasa u vezi kao i odgovarajući radni glagol.

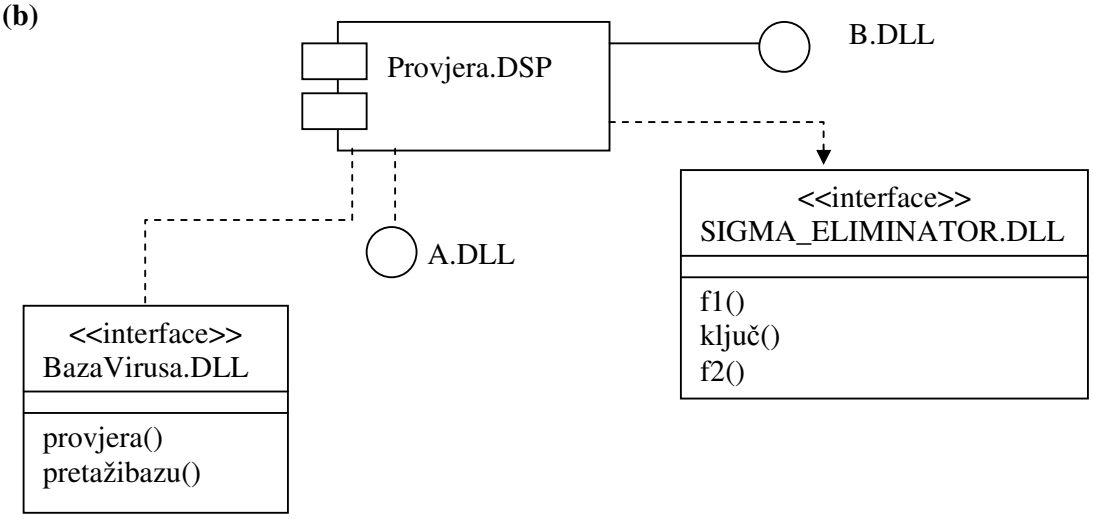
Rješenje:

(a)



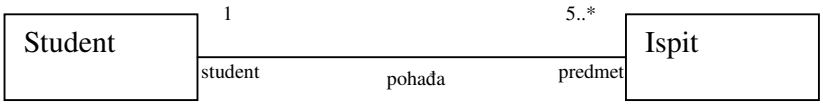
Riječ je o netipičnom pozicioniranju DMZ-a. Po pravilu DMZ se postavlja između terminala i rutera a na njega su vezani serveri SMTP, DNS, MAIL itd. Alternativno postoje dva DMZ-a jedan između terminala i servera a drugi između servera i spoljnog rutera. Za vježbu realizovati obiije ove varijante.

(b)



Napomena: Očigledno je moguće kreirati veliki broj ovakvih jednostavnih primjera sa komponentnama.

(c)

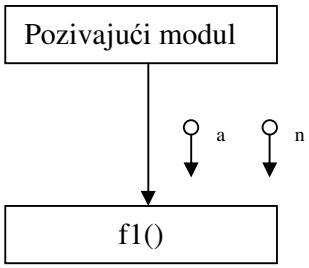


3. (a) Kreirati matricu ORGANIZACIJE/PROCESI. Procesi Pr1 i Pr2 su iz grupe Procesi planiranja, dok su procesi Pr3, Pr4 i Pr5 iz grupe Procesi proizvodnje. Direktor je glavni u procesima Pr1 i Pr3 dok je u ostale procese uključen. Organizaciona jedinica Or1 je glavna u procesu Pr2 dok je djelimično uključena u procese Pr4 i Pr5. Organizaciona jedinica Or2 je glavna u procesu Pr4 i uključena u proces Pr1.
(b) Program poziva funkciju koja je deklarisana kao: **void f1(const int *a,int n);** Nacrtati dijagram strukture modula i objasniti na koji su način povezani ovi moduli.

Rješenje.
(a)

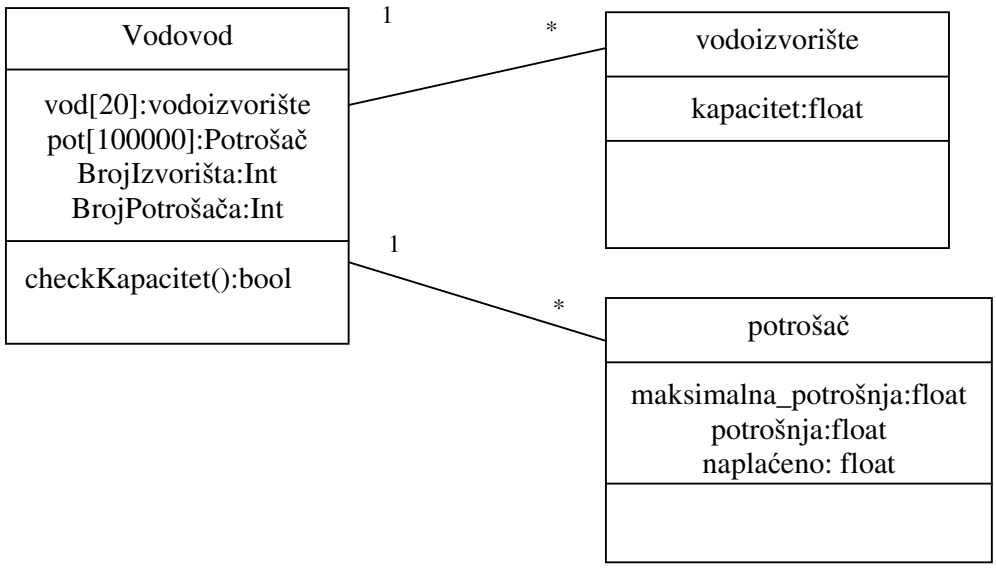
	Procesi planiranja		Procesi proizvodnje		
	Pr1	Pr2	Pr3	Pr4	Pr5
Direktor	x	/	x	/	/
Or1		x		z	z
Or2	/			X	

(b)



4. Kreirati klasu Vodovod. Klasa vodovod od podataka članova ima niz vodoizvorišta ne veći od 20. Svako vodoizvorište ima veličinu kapacitet. Pored toga klasa Vodovod ima niz Potrošača ne veći od 100000. Klasa Vodovod ima i dvije dodatne veličine BrojIzvorišta i BrojPotrošača. Svaki Potrošač ima veličinu maksimalna_potrošnja, veličinu potrošnja i veličinu naplaćeno. Kreirati ove tri klase i prikazati veze koje postoje između njih. Kreirati pseudokod funkcije koja određuje da li vodoizvorišta mogu da snadbiju potrošače sa traženim količinama vode (da li je suma kapaciteta svih vodoizvorišta veća od sume maksimalnih potrošnji prozvođača). Ako mogu funkcija treba da vrati rezultat 1 a ako ne mogu treba da vrati rezultat 0.

Rješenje:



Primjenili smo asocijativnu vezu Vodovoda sa vodoizvorištima i potrošačima. Razmislite da li je ovdje umjesto asocijativne veze mogla da se upotrijebi zavisnost za vezu prema jednoj odnosno drugoj. Objasnite vašu odluku.

Realizujmo sada funkciju koja provjerava kapacitet. Moguća realizacija:

```
Vodovovod::checkKapacitet():bool
i:Int
totalKapacitet:float
totalPotrošnja:float
totalKapacitet=0
totalPotrošnja=0
for i=1:BrojIzvorišta
totalKapacitet=totalKapacitet+vod[i].kapacitet
endfor
for i=1:BrojPotrošača
totalPotrošnja=totalPotrošnja+pot[i].maksimalna_potrošnja
endfor
if totalPotrošnja<totalKapacitet
return 1
else
return 0
endif
```

Realizacija je krajnje jednostavna ali postoji problem podaci članovi klasa vodovod i potrošač nisu vidljivu u klasi Vodovod osim ako ih ne deklarišemo kao javne promjenljive. Međutim, lako možete za vježbu kreirati getterse za navedene podatke i pristupati tim podacima putem ovih funkcija.

Napomena. Ovo je zadatak koji je u jednom terminu kolokvijuma došao sa nekim modifikacijama u više grupa. Razmislite što se moglo dopuniti odnosno dodati ovom zadatku.

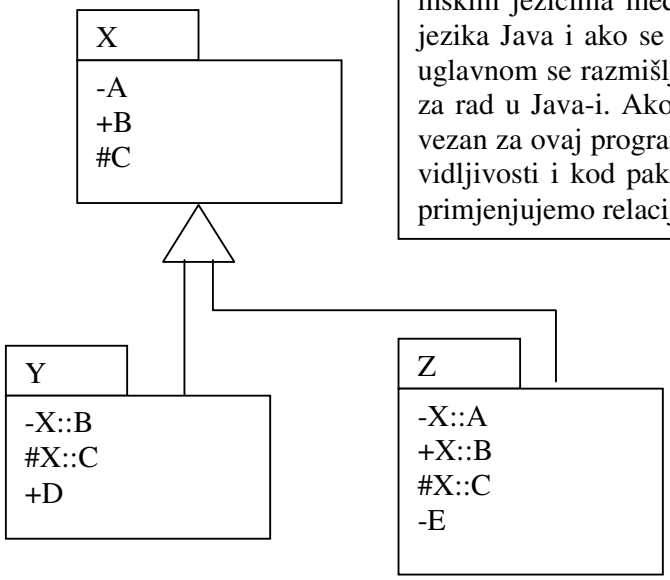
Jedan od primjera iz tog termina je bio i ovaj:

Kreirati klasu Vodovod. Klasa vodovod od podataka članova ima niz vodoizvorišta ne veći od 20. Svako vodoizvorište ima veličinu kapacitet. Pored toga klasa Vodovod ima niz Potrošača ne veći od 100000. Klasa Vodovod ima i dvije dodatne veličine BrojIzvorišta i BrojPotrošača. Svaki Potrošač ima veličinu maksimalna_potrošnja, veličinu potrošnja i veličinu naplaćeno. Kreirati ove tri klase i prikazati veze koje postoje između njih. Kreirati pseudokod funkcije koja određuje koliko ima potrošača koje treba isključiti sa vodovodne mreže. Kod njih je veličina potrošnja za više od 200 veća od veličine naplaćeno.

Uradite navedeni primjer za vježbu.

5. Prikazati vezu između paketa X koji sadrži elemente A (privatan), B (javan), C (zaštićen). Paket Y nasljeđuje paket X i iz njega preuzima B kojem mijenja vidljivost na privatan, C kojem zadržava vidljivost i dodaje svoju verziju elementa D (javan). Paket Z isto nasljeđuje paket X uz preuzimanje svih elemenata i dodaje elemenat E (privatan).

Rješenje:



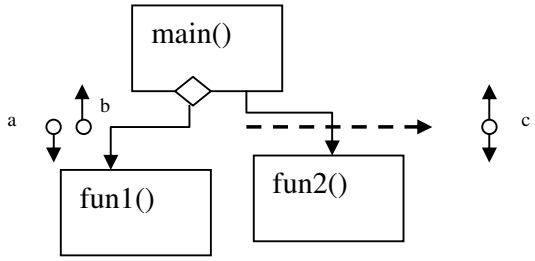
Paket je element koji na ovaj ili onaj način postoji u mnogim programskim jezicima međutim suštinski predstavlja kičmu programskog jezika Java i ako se radi sa paketima na način koji UML podržava uglavnom se razmišlja o programima i modelima koji su namjenjeni za rad u Java-i. Ako je to tako onda treba pogledati neki udžbenik vezan za ovaj programski jezik prije nego se odlučite za postavljanje vidljivosti i kod paketa koji se nasljeđuju a i kod paketa kod kojih primjenjujemo relaciju zavistnosti (import) – uvoženje.

6. Dat je dio program u programskom jeziku C (C++, Java):

```
#include <stdio.h>
int fun1(int);
void fun2(int *x);
main(){
…
if(){
…
a=fun1(b);
...
}
...
for(...;...;...) fun2(c);
...
}
```

Napisati DSM za ovaj program gdje su moduli glavni program i funkcije koje ovaj modul poziva (fun1 i fun2). Komentarisati povezanost ovih modula.

Rješenje:



Prvi modul je povezan podacima dok je drugi modul povezan preko zajedničke oblasti u memoriji (ili strukture podataka).

7. Kreirati klasu RAČUN koja ima šest podataka članova EURO, DINAR, KUNA, JEN, DOLAR i DATUM_PROMJENE. Prvih pet podataka članova su iznos na računu u odgovarajućoj valuti (ovaj broj može biti i negativan) dok je posljednji podatak datum posljednje primjene na računu. Napraviti po dva gettersa i settersa. Napisati metod koji računa koliko je stanje na računu u ekvivalentnoj valuti (Euro) tako što se uzima da je 1E=100 dinara, 1E=8 kuna, 1E=90 Jena i 1E=1.5 dolara. Napisati funkciju TUŽI koja ako je stanje na računu negativno za više od 1000E vraća 1 dok ako nije vraća 0.

Rješenje:

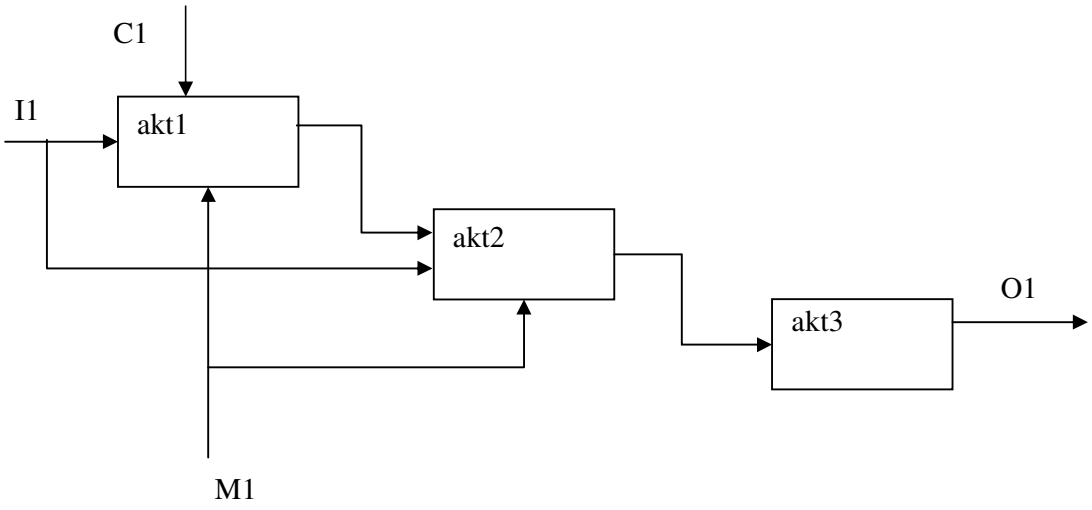
RAČUN
EURO:float DINAR:float KUNA:float JEN:float DOLAR:float DATUM_PROMJENE:Date
Ekviv():float TUŽI():bool

```
RAČUN::Ekviv():float
return EURO+DINAR/100+KUNA/8+JEN/90+DOLAR/1.5

RAČUN:TUŽI():bool
if Ekviv()<-1000
return 1
else
return 0
endif
```

8. Prikazati ICOM dijagram sa tri aktivnosti koje ćemo indeksirati sa akt1, akt2 i akt3. Prva aktivnost (akt1) ima ulaz I1, koji je ulaz i u aktivnost akt2 sa time da pored njega akt2 ima i ulaz koji je izlaz iz akt1. Aktivnost akt3 ima ulaz koji je izlaz iz akt2. Prva aktivnost ima kontrolu C1 dok su ostale aktivnosti bez kontrola. Prva i druga aktivnosti imaju isti mehanizam M1 dok je treća kontrola bez mehanizma. Izlaz iz treće aktivnosti je izlaz kompletnog sistema (O1).

Rješenje:

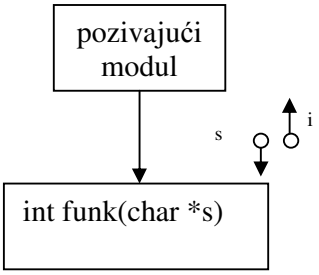


9. Informacioni sistem prekazan ICOM dijagramom u zadatku 8. Prva aktivnosti akt1 vezana za tri entiteta E1 (CRUD), E2 (CRU) i E3 (CU). Entitet E1 ima sljedeće atribute: a1 (IRUN), a2 (IRU), a3 (IRUN), a4 (IRUN), entitet E2 ima sljedeće atribute: a5 (IRU), a6 (IRUN), a7 (IUN), entitet E3 ima sljedeće atribute: a8 (IRUN), a9 (IRN), a10 (IRUN), a11 (IRUN), a12 (IRN). Druga aktivnosti ima dva entiteta E4 (CRUD) i E5 (CU). Entitet E4 ima sljedeće atribute b1 (IRUN), b2 (IRN), b3 (IUN) dok entitet E5 ima sljedeće atribute b4 (IRUN), b5 (IRUN), b6 (IRUN), b7 (R). U aktivnosti akt3 postoje identifikovani entiteti E6 (CRUD), E7 (CRD) i E8 (CRUD). Entitet E6 ima atribute: c1 (IRUN), c2 (IRU), c3 (IU), entitet E7 ima atribute: c4 (IRUN), c5 (IRUN), c6 (IRN) i entitet E8 ima atribute: c7 (R), c8 (IRUN), c9 (IRU) i c10 (IN). Kreirati CRUD-IRUN matricu za ovaj informacioni sistem.

Naziv aktivnosti	Naziv entiteta	CRUD	Naziv atributa	IRUN
akt1	E1	CRUD	a1	IRUN
			a2	IRU
			a3	IRUN
			a4	IRUN
	E2	CRU	a5	IRU
			a6	IRUN
			a7	I UN
	E3	C U	a8	IRUN
			a9	IR N
			a10	IRUN
			a11	IRUN
			a12	IR N
akt2	E4	CRUD	b1	IRUN
			b2	IR N
			b3	I UN
	E5	C U	b4	IRUN
			b5	IRUN
			b6	IRUN
			b7	..R
akt3	E6	CRUD	c1	IRUN
			c2	IRU
			c3	I U
	E7	CR D	c4	IRUN
			c5	IRUN
			c6	IR N
	E8	CRUD	c7	R
			c8	IRUN
			c9	IRU
			c10	I N

10. Da je kod sljedeće funkcije u programskom jeziku C:
int funk(char *s) {int i=0; while(s[i++]!='\0') ; return i;}
Prikazati vezu ove funkcije i modula koji je poziva. Kakva je povezanost ove dvije cjeline i koja je izmjena programskog koda potrebna da bi se povezanost popravila i da se ne izgubi ništa od funkcionalnosti modula.

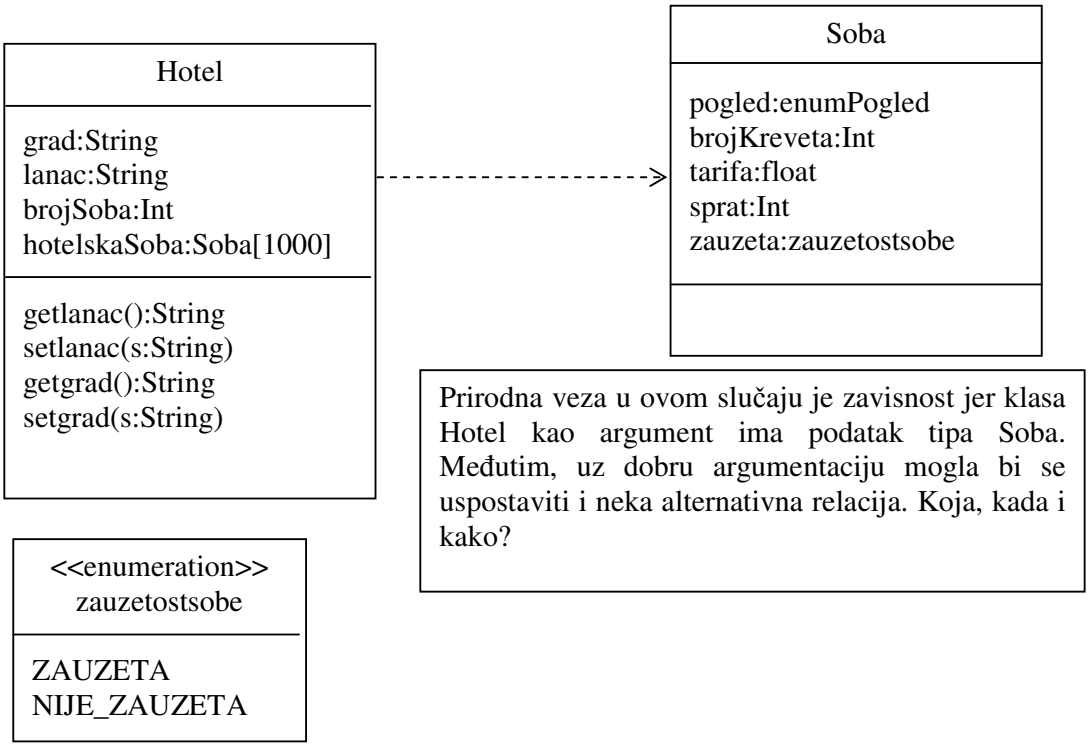
Rješenje:



Povezanost je u ovom primjeru loša jer su moduli povezani preko pokazivača s i promjene u njemu utiču na glavni program. Pošto pozivajući modul treba samo da vrati broj karaktera u stringu a ne i da mijenja string povezanost će se smanjiti (odnosno u našoj terminologiji) popraviti ako postavimo u zaglavlju funkcije
int funk(**const** char *s)

11. Kreirati klasu Hotel koja se sastoji od podataka o gradu, lancu kojem pripada hotel, broju soba, niza soba. Za ovu klasu imenovati na pravilan način atribute i dodijeliti im odgovarajuće tipove. Kreirati po dva gettersa i settersa. Kreirati klasu koja predstavlja apstrakciju sobe i koja ima podatke članove vezane za pogled, broj kreveta, tarifu, sprat i indikator da li je soba zauzeta ili nije zauzeta. Kreirati nabrajanje koje opisuje navedeni indikator. Prikazati vezu hotela i sobe i obrazložiti. Krerati fukciju i pravilno je pozicionirati u nekoj od navedenih klasa koja je cijena najskuplje sobe koja je trenutno u hotelu zauzeta.

Rješenje:



Prirodna veza u ovom slučaju je zavisnost jer klasa Hotel kao argument ima podatak tipa Soba. Međutim, uz dobru argumentaciju mogla bi se uspostaviti i neka alternativna relacija. Koja, kada i kako?

Funkcija NajskupljaIzdataSoba mora biti pozicionirana u klasi Hotel jer jedino ta klasa ima pristup podacima o sobama odnosno pojedinačna soba nema pristup podacima o ostalim sobama. Zaglavlje funkcije bi moralo biti:

Hotel::NajskupljaIzdataSoba():float

Nakon toga bi išle sljedeće naredbe:

Najskuplja:float
I:Int
Najskuplja=0

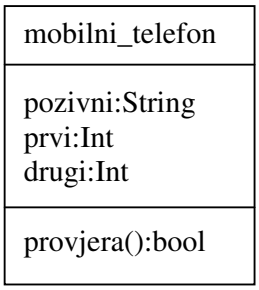
```
for I=1:brojSoba
        if hotelskaSoba[I].zauzeta==ZAUZETA && hotelskaSoba[I].tarifa>Najskuplja
        Najskuplja=hotelskaSoba[I].tarifa
        Endif
endfor

return Najskuplja
```

Predmetni program neće raditi! Zbog čega? Što treba izmjeniti da bi program radio? Da li se to može postići bez promjene vidljivosti elemenata klasa.

12. Kreirati klasu mobilni_telefon koja se sastoji od 3 podatka člana: stringa dužine 3, i dva cijela broja. Kreirati odgovarajuće konstruktore, destruktore, getterse i setterse odvojene sa odgovarajućim stereotipima. Dalje, napisati funkciju provjera koja provjerava da li je dati telefon ispravan. Da bi telefon imao ispravan broj string mora biti "067", "068" i "069" a brojevi moraju biti pozitivni najviše trocifreni.

Rješenje:

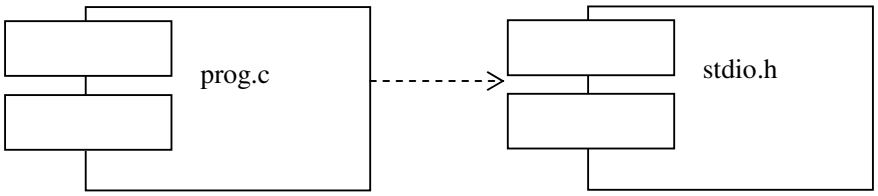


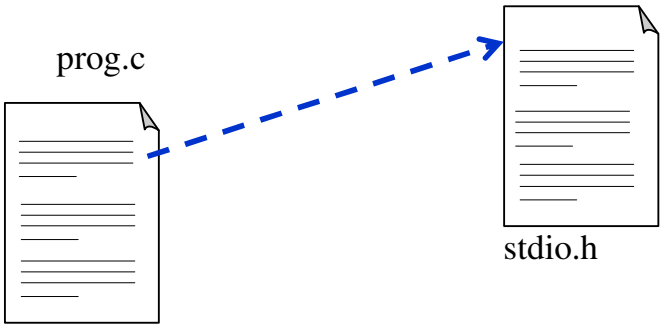

```
mobilni_telefon():bool
if (pozivni==“067“ || pozivni==“068“ || pozivni==“069“)&&
(prvi>=0&&prvi<=999)&&
(drugi>=0&&drugi<=999)
return true
else
return false
endif
```

Modifikujte ovaj zadatak tako što ćete u “067“ i “068“mreži zahtjevati da prvi broj bude najmanje 100 a u “069“ nemaovog ograničenja odnosno prvi broj mora biti veći ili jednak 10. Dozvoliti da u “067“ drugi broj bude četvorocifren ako je prvi član veći ili jednak 900.

13. Kojim elementom iz UML-a ćete modelovati programsku bibliteku stdio.h? Obrazložite i prikažite grafički.

Rješenje: Potpuno je ispravno da ovu programsku biblioteku modelujemo kao komponentu. Komponenta je kao što znamo fizički dio programskog koda u ovom slučaju i taj dio programskog koda se može koristiti u našim drugim programima. Na slici je prikazana veza programa i biblioteke koja koristi taj program. Alternativno kao na slici dolje za izvorni kod i biblioteku mogli smo da koristimo standardne stereotipe.





Konačno moguće je pretpostaviti da je stdio.h paket i da se pokaša kao paket pa bi se onda ovaj paket morao uključiti u programski kod putem zavisnosti sa stereotipom <<import>>. Iz više razloga ovo nije optimalno ni u potpunosti tačno ali je na nivou kojega mi izučavamo na našem kursu nešto što se može tolerisati.

14. Kreirati klasu Student koja pored uobičajenih podataka članova (ime, prezime, broj indeksa itd.) posjeduje i niz ocjena koje su dobijene na ispitima kao i cijeli broj koji označava koliko je student do sada položio ispita. Napisati osnovne getterse i setterse, konstruktore i destrukture. Ujedno kreirati i metod nagrada koji provjerava da li dati studente treba da dobije nagradu. Ako treba metod treba da vrati 1 a ako ne da vrati 0. Student zavrijeđuje nagradu ako ima najmanje 20 ocjena A i ne više od 2 C i niže ocjene.

Rješenje:

Student
Ime:String Prezime:String BrojIndeksa:String Ispiti[50]:Ocjene BrojPolIspita:Int
Student() ~Student() setIme(s:String) getIme():String setPrezime(s:String) getPrezime(s:String) nagrada():Int

Realizujmo jednu moguću varijantu konstruktora i destruktora:
Student::Student()
BrojPolIspita=0

Student::~Student()
I:Int
For I=1:BrojPolIspita
Ispiti[I]='F'
Endfor
BrojPolIspita=0

Realizacija mutatora:
setIme(s:String)
Ime=s

setPrezime(s:String)
Prezime=s

Realizacija inspektora:
getIme():String
return Ime

getPrezime():String
return Prezime

Realizacija metoda nagrada:
nagrada():Int
BrojA:Int
BrojC:Int
I:Int
BrojA=0
BrojC=0
For I=1:BrojPolIspita
If Ispiti[I]=='A'

BrojA=BrojA+1
ElseIf Ispiti[I]=='C' ∨ Ispiti[I]=='D' ∨ Ispiti[I]=='E'
BrojC=BrojC+1
EndIf
EndFor
If BrojA>=20 ∧ BrojC<=2
return 1
elseif
return 0
EndIf

Za vježbu sami realizujte enumeraciju koja predstavlja tip podatka ocjena. Napominjemo da je ovo takođe jedan od uobičajenih tipova zadataka koji se pojavljivao na našim kolokvijumima.

Pogledajmo sljedeću modifikaciju ovog zadatka:

Kreirati klasu Student koje ima sljedeće podatke ime, prezime, indikator upisa, godina upisa, broj položenih ispita. Napominjemo da je neophodno za atribute selektovati odgovarajuće tipove podataka i pravilna imena. Kreirati po dva gettersa i settersa. Kreirati i nabrajanje indikator upisa koje može da uzme vrijednosti UP_SAMOFINANSIRAJUCI, UP_REDOVNI, UP_UMANJENJE, UP_POLA. Kreirati funkciju koja provjerava status studenta. Ako je student položio manje od 7 ispita a upisan je kao UP_REDOVNI ili UP_UMANJENJE treba ga prebaciti na UP_SAMOFINASIRAJUCI dok ako je upisan kao UP_POLA i položio je više od 7 ispita treba ga prebaciti na status UP_REDOVNI. Kreirati klase Nastavnik i Ispit. Dodijeliti im barem po 3 podatka člana i prikazati i obrazložiti tipove veza između klase Student i ove dvije klase.

Rješenje:

Dajemo nekompletno rješenje isključivo da djelovima koji su bitno različiti u odnosu na prethodni primjer.

Unutar klase moramo da imamo promjenljivu:

indikator_upisa:INDIKATOR

Tip podataka INDIKATOR se realizuje putem nabrajanja:

Funkcija provjera_statusa() bi se mogla deklarisati baš na taj način u tijelu klase a realizovati na sljedeći način:

Student::provjera_statusa()
If BrojPolIspita<7 ∧(indikator_upisa==UP_REDOVNI∨ indikator_upisa==UP_POLA)
indikator_upisa=UP_SAMOFINANSIRAJUĆI
ElseIf BrojPolIspita>7 ∧ indikator_upisa==UP_POLA
indikator_upisa=UP_REDOVNI
EndIf

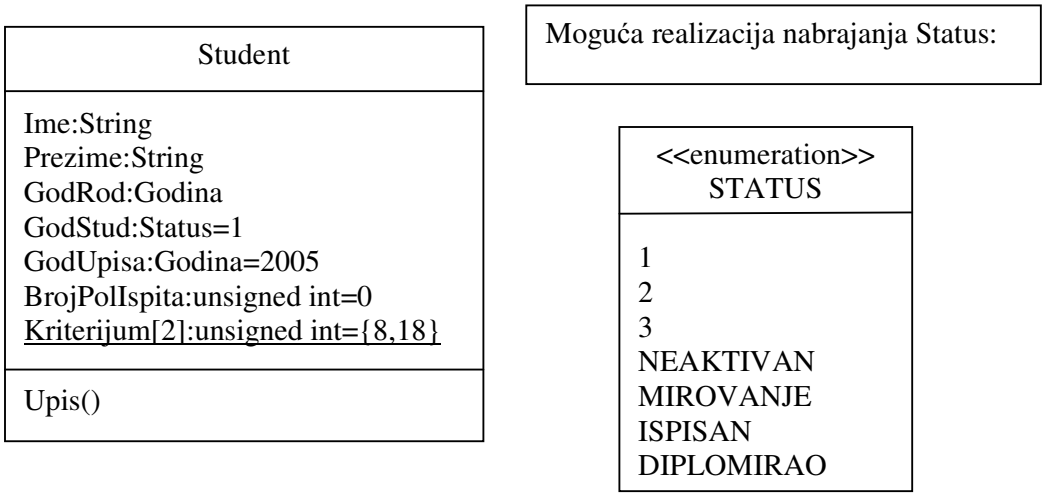
Što se tiče veze sa klasama Nastavnik i Ispit jasno je da klase Student i Nastavnik moraju svoju vezu koja se odvija preko klase Ispit. U svakom slučaju Nastavnik i Student su jaki entiteti koji sa Ispitom obavljaju vezu putem asocijativne relacije. Možda postoji izuzetak za slučaj kada postoji veza između Nastavnika i Studenta

kada je Nastavnik mentor Student-u na specijalističkom ili nekom sličnom radu. U ovom slučaju veza se može ostvariti sa studentom direktno, eventualno preko klase DiplomskiRad ili recimo preko odgovarajuće klase asocijacije.

Posmatrajmo sljedeću **modifikaciju** predmetnog **zadatka**.

Formirati klasu Student koja ima podatke članove: Ime (tipa String), Prezime (tipa String), GodRod (tipa Godina), GodUpisa (tipa Godina sa početnom vrijednošću 2005), GodStud (tipa Status sa početnom vrijednošću 1), BrojPolIsp (koji je tipa unsigned int i koji je na početku postavljen na 0), Kriterijum (koji ima dva člana koji su na početku postavljeni na 8 i 18 i koji su zajednički za čitavu klasu). Klasa ima funkcije članice za postavljanje vrijednosti promjenljivih članica kao i za čitanje njihovih vrijednosti. Klasa pored toga ima funkciju Upis koja upisuje studenta sa prve godine na drugu ako ima položeno više od broja ispita koji je prvi član niza Kriterijum a sa druge na treću ako student ima položeno više od broja ispita koji je drugi član niza kriterijum. Realizovati funkciju Upis i stereotip za tip podatka Status.

Rješenje. Ponovo ćemo dati samo skeč rješenja. Klasa (nisu prikazani gettersi i settersi) izgleda ovako:



Moguća realizacija metoda Upis() u ovom slučaju:

```
Student::Upis()
If GodStud≡1∧BrojPolIspita≥Kriterijum[1]
GodStud=2
ElseIf GodStud≡2∧BrojPolIspita≥Kriterijum[2]
GodStud=3
EndIf
```

15. Prikazati grafički (u skladu sa UML notacijom) apstraktnu klasu Radnik. Klasa ima osobine (atribute) datumZaposljenja tipa Datum, godineStaža (cijeli broj veći ili jednak nuli), maksGodineStaža (jedan atribut za čitavu klasu) po defaultu postavljen na 40, radnoMjesto (string dužine 10), ime (string dužine 10), prezime (string dužine 10), pol tipa karakter koji može uzeti vrijednost m ili ž. Svi podaci članovi su privatni. Klasa ima virtuelnu funkciju sračunajPlatu() koja vraća argument koji je tipa Euro, funkciju ažurirajStaž() koja uzima argument tipa Datum i računa staž osobe na osnovu tekućeg dana i dana zaposljenje, funkciju promjeniPrezime() koja uzima argument tipa string i mijenja prezime zaposlenog pod uslovom da je ženskog pola, i funkciju promjeniRadnoMjesto() koja uzima argument tipa string. Sve funkcije članice su javne. Dati kod, pseudokod ili algoritam bilo koje od funkcija članica klase.

Rješenje:

Radnik
datumZaposljenja:Datum godineStaža:UInt maksGodineStaža:Uint=40 radnoMjesto:char[10] ime:char[10] prezime:char[10] pol:char
sračunajPlatu():Euro ažurirajStaž(d:Datum) promjeniPrezime(c:char*) promjeniRadnoMjesto(c:char*)

Ovdje ćemo suštinski implementirati funkciju koja mijenja prezima uz napomenu da ovdje nemamo klasu String već koristimo pokazivač na karakter pa je i realizacija nešto drugačija (odnosno složenija). Napominjemo da oznaka char * predstavlja pokazivač na karakter:

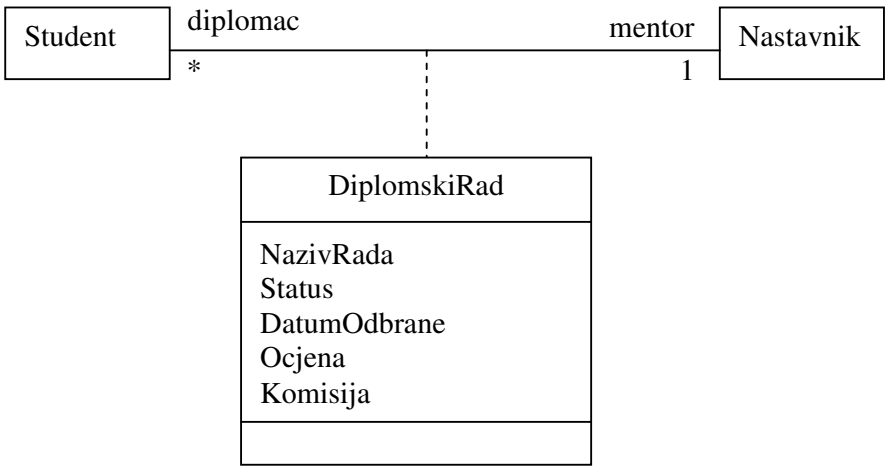
```
promjeniPrezime(c:char*)
I:Int
I=0
while c[I]≠'\0'
prezime[I]=c[I]
I=I+1
endwhile
prezime[I]='\0'
```

Napominjemo da je klasa Radnik takođe veoma pogodna za uvježbavanje i formiranje zadataka na kolokvijumima.

16. Veza Poručioca i Isporučioca robe je složena i modeluje se klasom asocijacije Faktura. Prikazati ovu vezu i za nadedenu klasu asocijacije prikazati nekoliko po vama bitnih podataka članova.

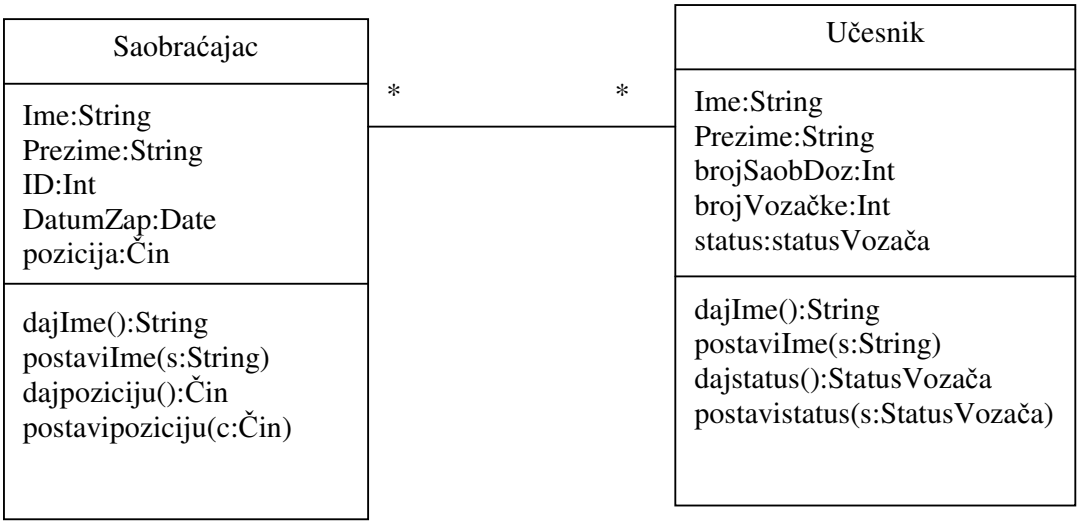
Rješenje: Predmetna veza je ilustovana na časovima predavanje i u prezentacijama. Stoga ćemo ovakav zadatak koji je dolazio na kolokvijumu zamjeniti sa sljedećom postavkom koja je već pomenuta u zadatku **14**.

Jedna od veza klase Student i klase Nastavnik u sistemu je vezana za diplomski rad. Ova veza je asocijativna ali je ujedno i dosta komplikovana pa se modeluje odgovarajućom klasom asocijacije.

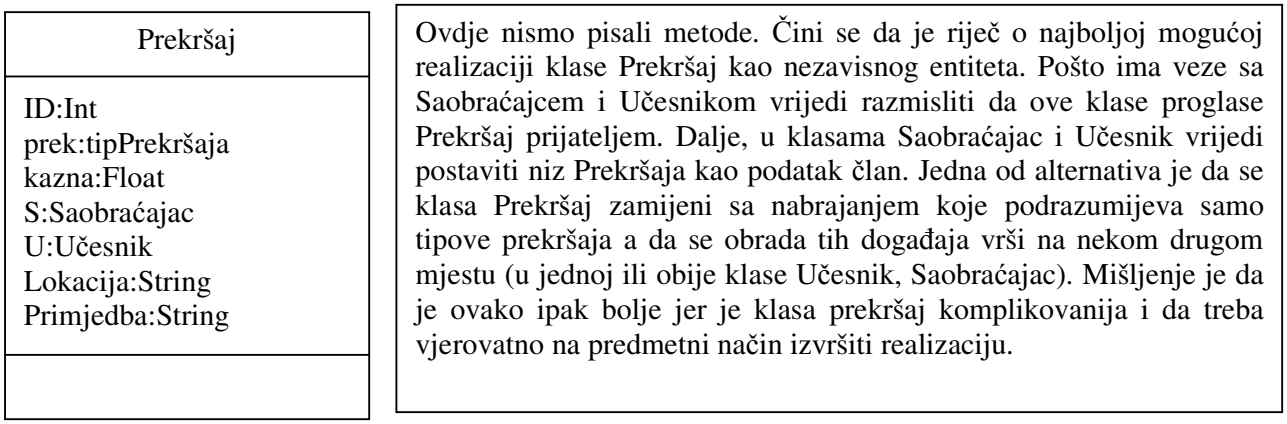


17. Kreirati klasu policajac koja treba da predstavlja apstrakciju službenika MUP-a zaduženog za poslove regulisanja i kontrole saobraćaja. Postavite mu barem 5 atributa kao i kreirajte barem 2 gettersa i settersa. Kreirati klasu koja predstavlja učesnika u saobraćaju. Opet mu dodijelite barem pet atributa i barem 2 gettersa i settersa. Kreirati i klasu Prekršaj koja označava saobraćajni prekršaj koji je počinio učesnik u saobraćaju a kojega je konstatovao policajac. Uspostavite veze između ovih klasa i razmotrite mogućnost da se klasa prekršaj pozicionira u jednoj od dvije prethodno opisane klase. Obrazložiti!

Rješenje:



Uz smišljavanje nabrajanja za Čin i statusVozača i pisanje funkcija za getterse i setterse ovim smo kompletirali prvi dio zadatka. Ostaje da se vidi kako realizovati klasu Prekršaj. Jedna moguća realizacija bi mogla da bude:

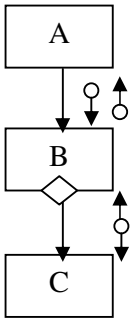


Za vježbu smislite način kako bi u predmetnom sistemu bilo moguće naplatiti kaznu, provjeriti evidenciju o kaznama učesnika itd.

18. (a) Prikazati DTP na kojem modul A poziva modul B a ovaj uslovno modul C. Dodati atribute tako da povezanst modula A i B bude normalna a povezanost modula B i C bude povezanost podacima. Odrediti FAN IN i FAN OUT za ove module.
(b) Prikazati matricu PROCESI-ORGANIZACIJE. Procesi pripadaju grupama STRATEŠKO PLANIRANJE (procesi P1, P2 i P3) i MARKETING (procesi P4, P5). Organizacione jedinice i pojedinci u organizaciji su: upravni odbor (glavni u procesu P1, uključen u P4 i P5, djelimično uključen u P2 i P3); izvršni direktor (glavni u procesima P2 i P3, uključen u P4, djelimično uključen u P1); marketing služba (glavna u procesu P4, uključena u P5 i djelimično uključena u P1 i P3); računovodstvo (uključeno u procese P2, P4 i P5).

Rješenje:

(a)

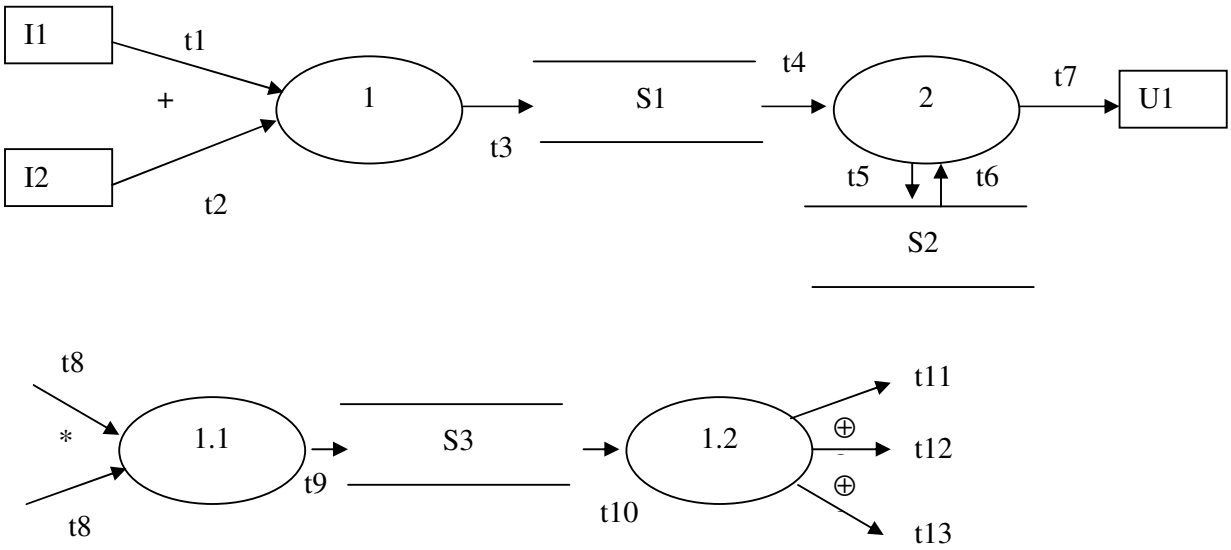


A FAN IN=0 FAN OUT=1
B FAN IN=1 FAN OUT=1
C FAN IN=1 FAN OUT=0

(b) Za rješenje ovog zadatka pogledajte zadatak tri iz ovog dodatka.

19. Prikazati DTP na kojem podaci sa izvora I1 i I2 (dovoljan je jedan tok ali se mogu pojaviti i oba istovremeno) dolaze do procesa 1 koji preko skladišta S1 prosljeđuje u proces 2. Proces 2 komunicira sa skladištem S2 a rezultate rada smiješta u uvir U1. Imenovati sve tokove podataka na ovom dijagramu. Proces 1 je dekomponovan na procese 1.1 i 1.2. 1.1 prima podatke preko dva toka (moraju postojati oba) i preko skladišta S3 prosljeđuje procesu 1.2. Ovaj proces ima tri izlazna toka podataka od kojih može biti prisutan samo jedan. Imenujte sve tokove podataka i na ovom dijagramu i formulišite pravilo balansa za proces 1.

Rješenje:



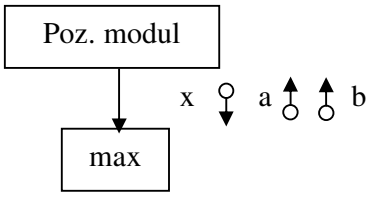
Pravilo balan sa za ulazne tokove podataka:
t1∪t2=t8∪t9
Pravilo balansa za izlazne tokove podataka:
t3=t11∪t12∪t13

20. (a) MATLAB funkcija **[a,b]=max(x)** određuje maksimum niza a i poziciju maksimuma b. Prikazati DTP veze ova dva modula.
(b) MATLAB funkcija **[a,b]=sort(x)** određuje sortirani niz a i indekse sa pozicijama sortiranih elemenata u osnovnom nizu. Prikazati DTP veze ova dva modula.
(c) Proces izdavanja novca na automat se obavlja na osnovu podataka sa bankinog računa, kao i podataka koje unese korisnik. Podaci sa bankinog računa su u obliku skladišta podataka a ne predstavljaju aktivni izvor ni uvir

podataka. Nakon obrade rezultati se uvijek skladište. Pored toga rezultat ide kao obavještenje na ekran (uvijek) ali može i u vidu priznanice ili u vidu novca.

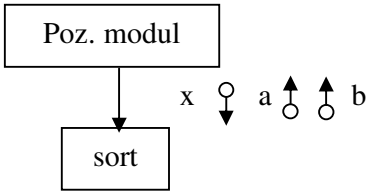
Rješenje:

(a)



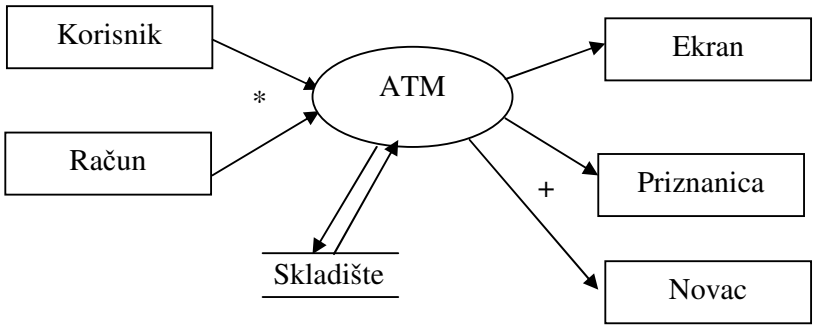
U pitanju je povezanost podacima!

(b)



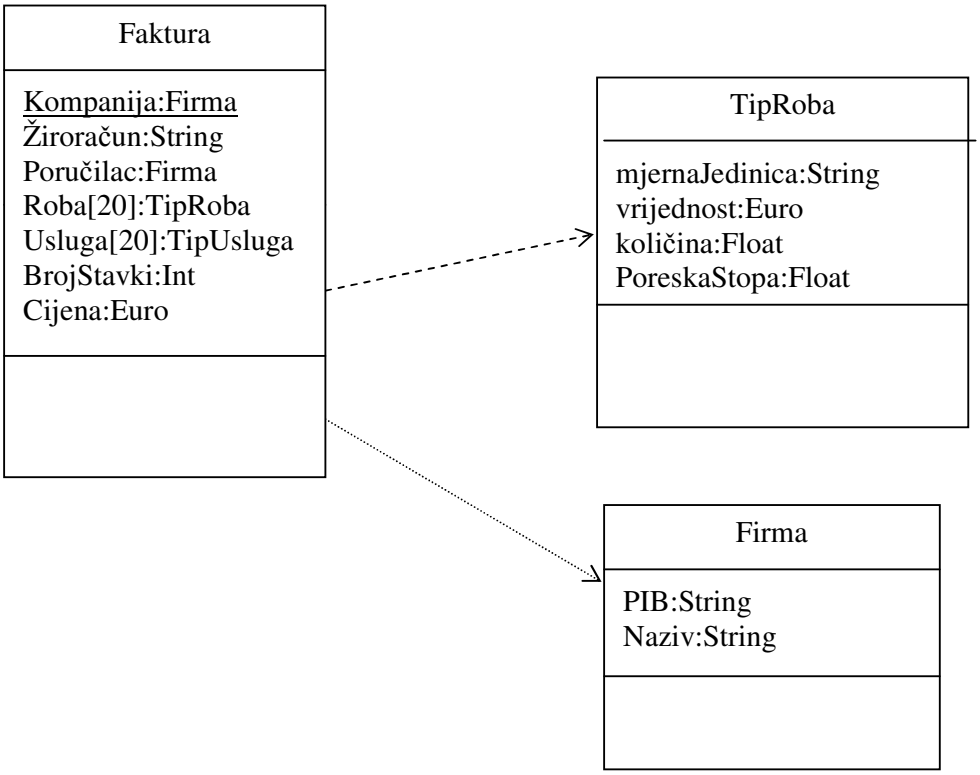
U pitanju je povezanost podacima!
Niz se ponaša kao pravi podatak i izmjene na njemu ne utiču na glavni program. Ovakvo razdvajanje u programskom jeziku C nije moguće a u C++ jeste ali kako?

(c)



21. Formirati klasu podataka Faktura. Podaci koje sadrži su vezani za to ko fakturiše, žiroračun, kome fakturiše i koju robu ili uslugu fakturiše, koliko se stavki fakturiše i ukupnu cijenu fakture. Klasa treba da ima podatke koji odražavaju ove elemente fakture kao i osnovne metode koji sa time barataju. Pretpostavka je da svaka Faktura ima isti podatak vezan za "ko fakturiše" i da je to zajednički podatak za čitavu klasu. Vezano za robu ili usluge kreirati odgovarajuću klasu (koja će biti korišćena kao tip podatka u klasi Faktura). Osnovni podaci u toj klasi su: mjerna jedinica, vrijednost po mjernoj jedinici, količina, naziv, vrijednost poreza koji se zaračunava za tu robu. I ova klasa mora da ima osnovne metode koji očitavaju i mijenjaju stanje pojedinih podataka članova. Kreirajte i klasu Naziv pomoću koje će kao tipa podataka biti modelovani podaci "ko fakturiše" i "kome fakturiše" sa odgovarajućim inspektorima i mutatorima. Napisati funkciju koja pomoću zadatih podataka (cijene proizvoda, fakturisane količine i poreza na pojedine robe ili usluge) računa i postavlja ukupnu vrijednost fakturisane robe.

Rješenje:



Nismo implementirali jednostavne metode (konstruktor, destruktor, getterse i setterse) pošto smo mišljenja da ih ne treba posebno objašnjavati na ovom nivou. Opredijelili smo se za relaciju zavisnosti pošto se navedeni tipovi pojavljuju u klasi Faktura. Ostatak zadatka vam prepuštam.