Dr Milunka Damnjanović, red.prof,
OBJEKTNO ORIJENTISANE TEHNIKE PROJEKTOVANJA SISTEMA

11 Projektovanje objekata

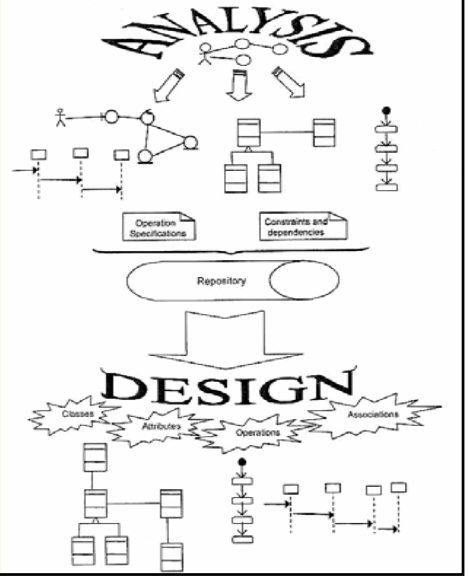
Zadatak projektovanja objekata #1:

- **Dizajn objekta je povezan sa detaljnim dizajniranjem objekata i njihovih interakcija.**
- **Generalno pravilo je da se promene na strukturama koje su specificirane za vreme analize svedu na minimum.**

Zadatak projektovanja objekata #2:

- **Veliki deo dizajnerske aktivnosti fokusira se na dodavanje detalja specifikaciji analize (tipova atributa, kako funkcionišu operacije i kako su objekti linkovani sa drugim objektima) što ne stvara promenu na strukturama koje su identifikovane za vreme analize. (Medjutim, može postojati potreba da se strukture analize modifikuju iz različitih razloga.)**
- **Dizajn objekta proizvodi detaljnu specifikaciju klasa koristeći notaciju UML-a.**

Izvori informacija za projektovanje objekata:



Specifikacija klase - ATRIBUTI:

Osnovni tipovi podataka:

- **Bulov (boolean, *true* i *false*),**
- **karakter (character, *svi alfanumerici i specijalni karakteri*),**
- **celobrojni (integer, *celi brojevi*) i**
- **decimalni (floating-point, *decimalni brojevi*).**

Sintaksa tipa atributa:

Tip podatka atributa u UML-u deklariše se sledećom sintaksom:

```
ime ':' izraz-tipa '=' inicijalna-vrednost
                        '{'string-svojstava'}'
```

gde je: `ime` **– ime atributa,**
`izraz-tipa` **– njegov tip,**
`inicijalna-vrednost` **- vrednost atributa kada se objekat kreira prvi put,**
`string-svojstava` **- opisuje svojstva atributa (kao konstantu ili fiksnu vrednost, npr.)**

Specifikacija klase – primer:

Bankovni racun
Brojracuna:celi brojevi Imeracuna:red{nije nula} ravnoteza:novac=0 dostupnaravnoteza:novac granicaprekoracenja:novac
Otvoren(imeracuna:red):Boolean zatvoren():Boolean Kredit(iznos:novac):Boolean Isplata(iznos:novac):Boolean Dobitiravnotezu():novac Postavitiravnotezu(novaravnoteza:novac) Dobitiimeracuna():red Postavitiimeracuna(novoime:red)

Specifikacija klase - OPERACIJE:

Svaka operacija mora biti specificirana pomoću parametara koje prenosi i vraća.

Sintaksa je:

```
ime operacije '('lista-parametara ')'':'
                          izraz-tipa-vraćenog
```

Specifikacija klase – primer OPERACIJE:

Klasa `Bankovniracun` **moze imati operaciju** `kredit()` **koja prenosi kreditirani iznos do prijemnog objekta i ima povratnu vrednost tipa Boolean. Operacija može biti definisana koristeći sintaksu:**

`kredit(iznos:novac):Boolean`

poruka koju `kredit()` **šalje objektu** `Bankovniracun` **moze imati oblik:**

`kreditOK=objekatracuna.kredit(500.00),`

gde `kredit ok` **zadrzava Boolean povratnu vrednost koja je dostupna prijemnom objektu kada operacija** `kredit()` **zavrsi rad.**

Specifikacija klase – VIDLJIVOST OBJEKTA #1:

U toku analize prave se razne pretpostavke o granici inkapsulacije objekta i načinu interakcije sa drugim objektima.

Prelaskom na projekovanje, donose se odluke koje se odnose na to koje operacije (i možda atributi) su javno dostupni. Dakle, definiše se granica inkapsulacije.

Specifikacija klase – VIDLJIVOST OBJEKTA #2:

Klasa `Bankovniracun` **sa specifičnim tipovima osobina i definisanim parametrima operacije ima osobinu** `ravnoteza` **sto se može videti tokom analize, tako da ona može biti pokrenuta direktno uz pomoć jednostavnih primarnih operacija** `Dobitiravnotezu()` **i** `Posticiravnotezu()`**. Medjutim** `ravnoteza` **treba da bude inovirana operacijama** `kredit() i isplata()`**, jer se promene vrednosti osobine** `ravnoteze` **mogu samo javiti tokom operacije** `isplata()` **i** `kredit()`**.Operacija** `posticiravnotezu()` **ne treba biti javno dostupna na korišćenje drugim klasama.**

Specifikacija klase – VIDLJIVOST OBJEKTA #3:

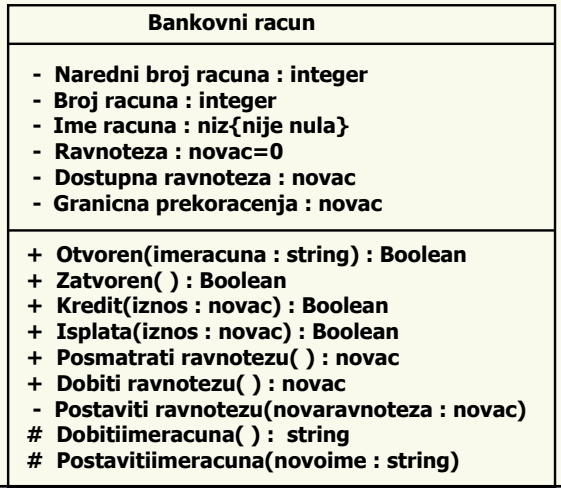
Četiri uobičajena termina koji su usvojeni da opišu vidljivost su:

simbol vidljivosti	vidljivost	značenje
+	javna	Karakteristika (operacija ili osobina) je direktno vidljiva u svakoj klasi.
-	privatna	Karakteristika može jedino biti upotrebljena na uzorku klase kojom je obuhvaćen.

Specifikacija klase – VIDLJIVOST OBJEKTA #4:

simbol vidljivosti	vidljivost	značenje
#	zaštićena	karakteristiku mogu da koriste uzorci klase koja je uključuje ili potklasa date klase.
~	paket	Karakteristika je direktno dostupna samo uzorcima klase u istom paketu.

Specifikacija klase – primer specifikacije vidljivosti objekta:



Interfejsi:

- Ponekad jedna klasa (ili neka druga komponenta) mogu da prikažu više od jednog interfejsa drugim klasama ili se jedan isti interfejs može zahtevati od više klasa.
- U UML-u, interfejs je spolja vidljiva (t.j. javna) operacija.
- Interfejs ne sadrži unutrašnju strukturu, nema atribute ni asocijacije i izvodjenje operacija nije definisano.
- Formalno, interfejs je ekvivalent apstraktnoj klasi koja nema atribute, ni asocijacije već samo apstraktne operacije.

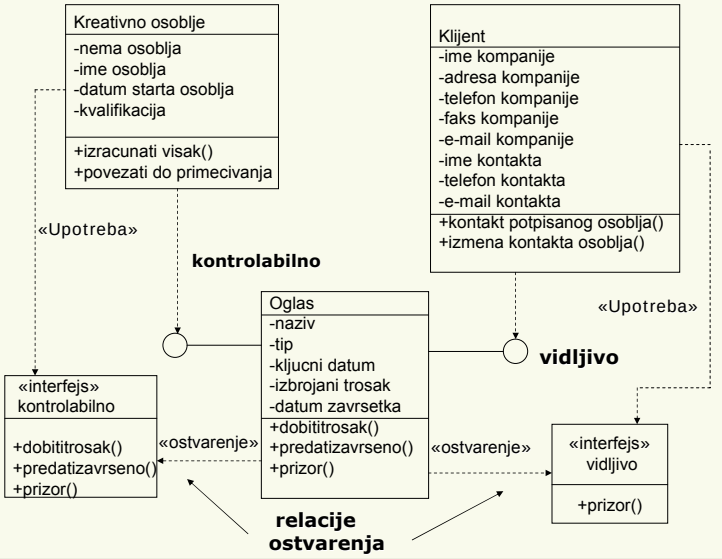
Primer opisa i upotrebe interfejsa #1:

- Na sledećoj slici prikazana su dva alternativna sistema znakova za interfejs: Klasa `oglas` podržava dva interfejsa `kontrolabilno` i `vidljivo`. Kružni sistem znakova ne obuhvata listu operacija obuhvaćenih tipom interfejsa i ako one treba da budu nabrojane u skladištu. Isprekidana linija u klasi `kreativnoosoblje` do kružića ikone interfejsa `kontrolabilno` ukazuje da ona koristi, ili su joj potrebne u većini slučajeva, operacije koje obezbeđuje interfejs.

Primer opisa i upotrebe interfejsa #2:

Alternativni sistem znakova koristi stereotipnu klasnu ikonu. Ovaj sistem znakova navodi operacije na dijagramu.

Veza ostvarenje (predsstavljena isprekidanom linijom i trouglićem ističe da klasa klijenta(npr.`oglas`)podržava najmanje one operacije koje su nabrojane u interfejsu (npr.`kontrolabilno` ili `vidljivo`). Zatim isprekidana strelica iz `kreativnogosoblja` znači da su klasi potrebne ili koristi samo operacije koje su navedene u interfejsu. `Oglas` nasleđuje interfejs `kontrolabilno`.



Kriterijumi za dobar projekat - sjedinjavanje:

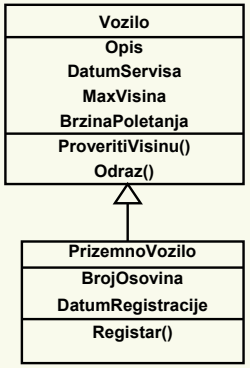
Sjedinjavanje **opisuje stepen unutrasnje povezanosti izmedju komponenti dizajna i ogleda se brojem veza koje objekat sadrži i stepenom interakcije objekta sa ostalim objektima.**

Kohezija **je mera stepena do kojeg elemenat služi jednoj nameni.**

Sjedinjavanje interakcije **je mera broja tipova poruka koje jedan objekat pošalje drugim objektima ii broj parametara koji se prenose sa ovim tipovima poruka. Trebalo bi ga držati na minimumu da bi se olakšala komunikacija i ponovna upotreba.**

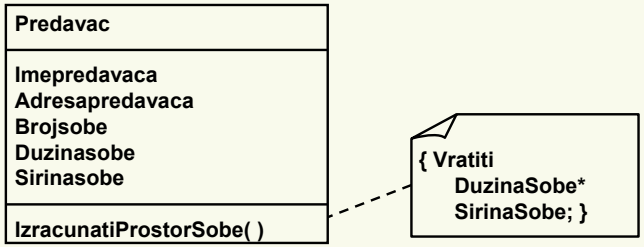
Nasledno sjedinjavanje:

Nasledno sjedinjavanje **opisuje stepen do kojeg su potklasi potrebne karakteristike koje nasleđuje iz klasne osnove. Potklasi nisu potrebne osobine `MaxVisina` i `BrzinaPoletanja` niti operacije `ProveritiVisinu()` i `Odraz()`. One su nepotrebno nasleđene. U ovom primeru će se javiti klasna osnova `Vozilo` kojoj bi bolje pristajalo ime `LetećeVozilo`. Međutim, potklasa sa nepotrebnim osobinama i operacijama je kompleksnija nego što treba da bude i objekti potklase zauzimaju više memorije nego što im je stvarno potrebno. Zato, nepotrebno nasledjivanje treba smanjiti koliko god je to moguce.**



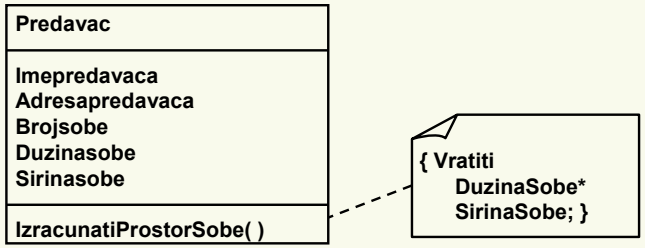
Kriterijumi za dobar projekat – kohezija operacija:

Kohezija operacija **meri stepen do koga se jedna operacija fokusira na jedan funkcionalni zahtev. Dobar projekat ima visoko kohezione operacije od kojih svaka radi sa jednim funkcionalnim zahtevom. Operacija `IzracunatiProstorSobe( )` je visoko koheziona.**



Kriterijumi za dobar projekat – kohezija klasa:

Kohezija klasa **odražava stepen do koga je klasa fokusirana na jedan zahtev. Klasa `Predavac` ima mali stepen kohezije jer ima tri atributa (`Brojsobe, Duzinasobe` i `Sirinasobe`) i jednu operaciju (`IzracunatiProstorSobe( )`) što bi više odgovaralo klasi `soba`.**

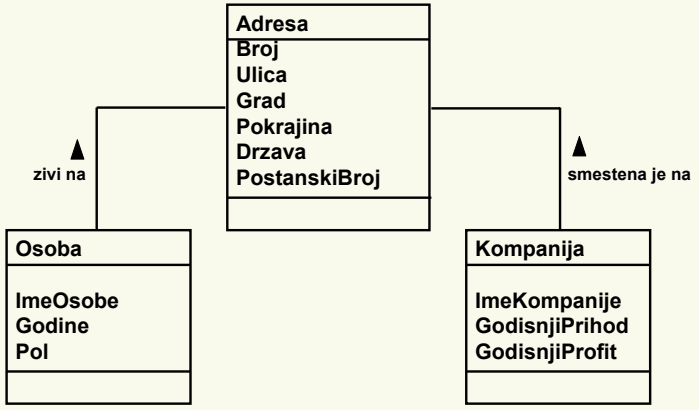


Kriterijumi za dobar projekat – kohezija specijalizacije:

Kohezija specijalizacije **odnosi se na koheziju hijerarhije nasleđivanja.**

Na slici je prikazan primer visokog naslednog sjedinjavanja, ali niske kohezije specijalizacije jer ni `Osoba` ni `Kompanija` nisu vrsta adrese.

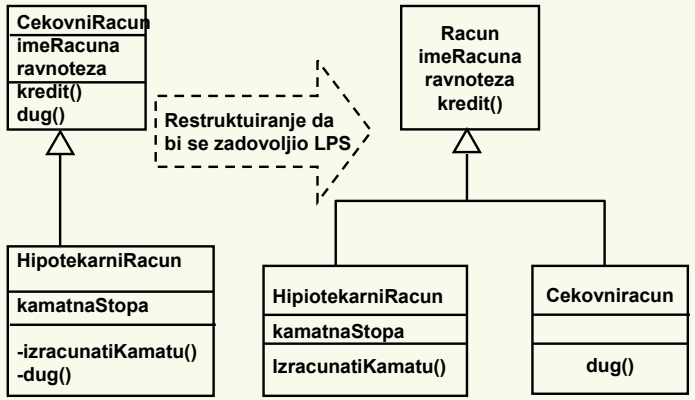
Primer bolje kohezije specijalizacije:



Liskov princip supstitucije:

- *Liskov princip supstitucije (LPS)* **je drugi kriterijum dobrog dizajna koji se primenjuje na hijerarhiju nasleđivanja. Ovaj princip kaže da u interakciji objekata, proizvedeni objekat može se tretirati kao da je osnovni objekat. Na slici 14.7 objekti klase kreditniracun() se ne mogu tretirati kao objekti klase cekovniracun() zato sto objekti kreditniracun() nemaju operaciju isplatadok objekti cekovniracun() imaju.Operacija isplata je privatna operacija u kreditniracun i ne moze se iskoristiti ni za jedan drugi objekat.**

Aplikacija Liskovog principa supstitucije:



Jasnost dizajna:

Dizajn treba da bude napravljen tako da je što razumljiviji, sto znači da ne treba preterivati sa dodavanjem nepotrebnih mogućnosti dizajnu.

- **Poruke i operacije treba da su sto prostije.**
- **Dobar dizajn mora da bude stabilan bez obzira na promene u zahtavima. To se postiže forsiranjem inkapsulacije.**
- **Kompleksan objekat treba rastaviti na delove (ako je to moguce), tj. na komponente objekta koji formiraju kompoziciju ili sjedinjavanje.**

Projektovanje asocijacije #1:

Asocijacija izmedu dve klase predstavlja mogućnost linkovanja izmedju klasa.

Linkovi omogućavaju veze koje su neophodne da se pojave poruke.

Postoji više tipova asocijacija :

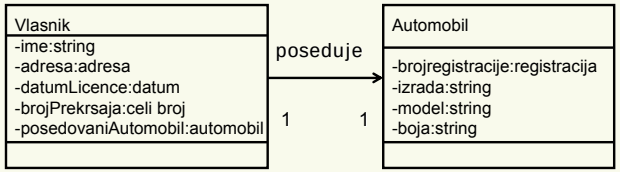
- **asocijacija jedan-na-jedan**
- **asocijacija jedan-na-više**
- **asocijacija više-na-više**

Asocijacija jedan-na-jedan:

Na slici objekti klase `Vlasnik` treba da šalju poruke objektima klase `Automobil` ali ne i obrnuto.

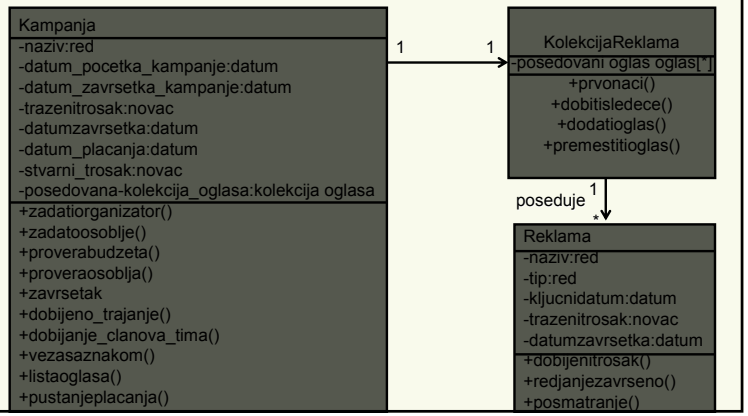
Objekti `Vlasnik` imaju identifikatore objekta `Automobil` i mogu da šalju poruke linkovanom objektu `Automobil`.

S obzirom da objekat `Automobil` nema identifikator za objekat `Vlasnik`, ne može slati poruke objektu `Vlasnik`.



Asocijacija jedan-na-više:

Vise identifikatora objekta `Reklama` treba da budu povezani sa jednim objektom `Kampanje` otuda i naziv asocijacija jedan-na-više.



Asocijacija više-na-više:

S obzirom da je ovo dvosmerna asocijacija svakom objektu `Kampanje` su potrebni identifikatori objekta `Kreativnoosoblje` i samom objektu `Kreativnoosoblje` treba kolekcija identifikatora objekta `Kampanja`. Na slici je prikazana dvosmerna vise-po-vise asocijacija i klasa `KreativnoOsoblje` i `Kampanja` sadrže identifikatore objekta njihovih kolekcijskih klasa.

Projektovanje operacija #1:

Projekat operacije uključuje određivanje najboljeg algoritma za izvođenje zahtevane funkcije.

Različiti faktori ograničavaju dizajn algoritma:

- **cena implementacije,**
- **ograničenja osobina,**
- **zahtevi za tačnost i**
- **sposobnosti implementacione platforme.**

Generalno, treba odabrati najjednostavniji algoritam koji zadovoljava ograničenja jer to čini operaciju lakše izvodivom i lakšom za održavanje.

Projektovanje operacija #2:

Pri izboru alternativnih projekata algoritama za operacije, treba razmotriti faktore kao:

Računarska kompleksnost **– zahtevano vreme i količina memorije.**

Lakoća implementacije i razumljivost **– generalno je bolje žrtvovati neku osobinu da bi se pojednostavila implementacija.**

Fleksibilnost **– treba imati na umu potencijalne izmene.**

Fino podešavanje modela objekta **može pojednostaviti algoritam.**

Normalizacija:

Za dva atributa A i B , A je funkcionalno zavisno od B ako za svaku vrednost B postoji tačno jedna vrednost A asocirana sa njim u bilo koje dato vreme. Notacija za ovo je : B → A .

Atributi mogu biti grupisani oko fukcionalnih zavisnosti, u skladu sa nekoliko pravila normalizacije, da bi se proizvele normalizovane strukture podataka koje nemaju redundansu.

Normalizacija može biti korisna kada se koriste relacione baze podataka kao deo implemntacione platforme ili kao vodič za dekompoziciju velikog kompleksnog (i verovatno ne jako kohezionog) objekta.

Većina OO pristupa razvoju softvera ne smatraju normalizaciju neophodnom i strukture koje nisu normalizovane smatraju prihvatljivim.

Generalno, OO pristupi ako su primenjeni sa odgovarajućim ograničenjima kvaliteta, proizvešće strukture bez redundanse.

Kratak pregled:

Detaljni proces projektovanja uključuje:

- **tip podataka atributa,**
- **specifikaciju interfejsa i definisanje signatura operacija,**
- **projekat asocijacija,**
- **referenciranje objekata u klasama,**
- **ograničenja integriteta,**
- **seriju kriterijuma koji obuhvataju sjedinjavanje i koheziju,**
- **normalizaciju klasa.**