

# DIJAGRAMI INTERAKCIJE

- Interakcija je ponašanje koje obuhvata skup poruka koje se razmenjuju između skupa objekata u nekom kontekstu sa nekom namenom
- Poruka je specifikacija komunikacije između objekata koja prenosi informaciju
- Od poruke se očekuje da će uslediti aktivnost
- Interakcija se koristi za modeliranje dinamičkih aspekata modela



## VRSTE DIJAGRAMA INTERAKCIJE

- Dva prikaza interakcije daju dve izomorfne vrste dijagrama interakcije
  - **Dijagrami sekvence**
  - **Dijagrami kolaboracije**
- Dijagrami sekvence naglašavaju **vremensko** uređenje interakcije
- Dijagrami kolaboracije naglašavaju **strukturu** veza između učesnika u interakciji
- Ove dve vrste dijagrama vizuelizuju na različit način iste informacije
- Semantički su potpuno ekvivalentni i mogu se automatski konvertovati jedan u drugi



# KONTEKST

- Interakcija se koristi za modeliranje toka kontrole, ali u različitim kontekstima
- Kontekst može biti:
  - sistem ili podsistem
  - operacija
  - klasa
  - slučaj korišćenja
- Kontekst – sistem ili podsistem kao celina
  - interakcije su u kolaboraciji objekata koji postoje u sistemu ili podsistemu
  - Primer: sistem za Web-commerce: sarađuju objekti na strani klijenta sa objektima na strani servera



# KONTEKST

- Kontekst – operacija
  - interakcije su među objektima koji implementiraju operaciju
  - parametri operacije, lokalni i globalni objekti mogu interagovati da izvrše algoritam operacije
- Kontekst – klasa
  - atributi klase mogu kolaborirati međusobno kao i sa globalnim objektima i parametrima operacija
  - interakcija se koristi da opiše semantiku klase
- Kontekst – slučaj korišćenja
  - interakcija reprezentuje scenario za slučaj korišćenja



# OBJEKTI I NJIHOVE ULOGE (1)

- Objekti koji učestvuju u jednoj interakciji mogu biti:
  - konkretne stvari
  - prototipske stvari
- Konkretne stvari su stvari iz realnog sveta
  - na primer, `o` kao instanca klase `Osoba` može označavati konkretnu osobu
- Prototipske stvari su stvari koje označavaju proizvoljnu stvar nekog tipa
  - na primer, `o` kao instanca klase `Osoba` može reprezentovati proizvoljnu osobu
- U kolaboracijama
  - objekti su prototipske stvari koje igraju specifične uloge
  - objekti su specifične instance iz realnog sveta



## OBJEKTI I NJIHOVE ULOGE (2)

- U interakciji se mogu pojaviti "instance" apstraktnih klasa i interfejsa
  - ovde instance ne označavaju konkretne stvari (nemoguće su direktne instance)
  - ovde instance reprezentuju prototipske stvari (instance subklasa)
- U kontekstu interakcije postoje instance: klasa, komponenata, čvorova i slučajeva korišćenja
- Objektni dijagram specificira objekte koji sarađuju i veze između njih
- Objektni dijagram je reprezentacija statičkih aspekata interakcije
- Interakcija uvodi dinamički aspekt specificirajući sekvencu poruka koje razmenjuju objekti
- Slanje poruke objektu predstavlja poziv adekvatne operacije



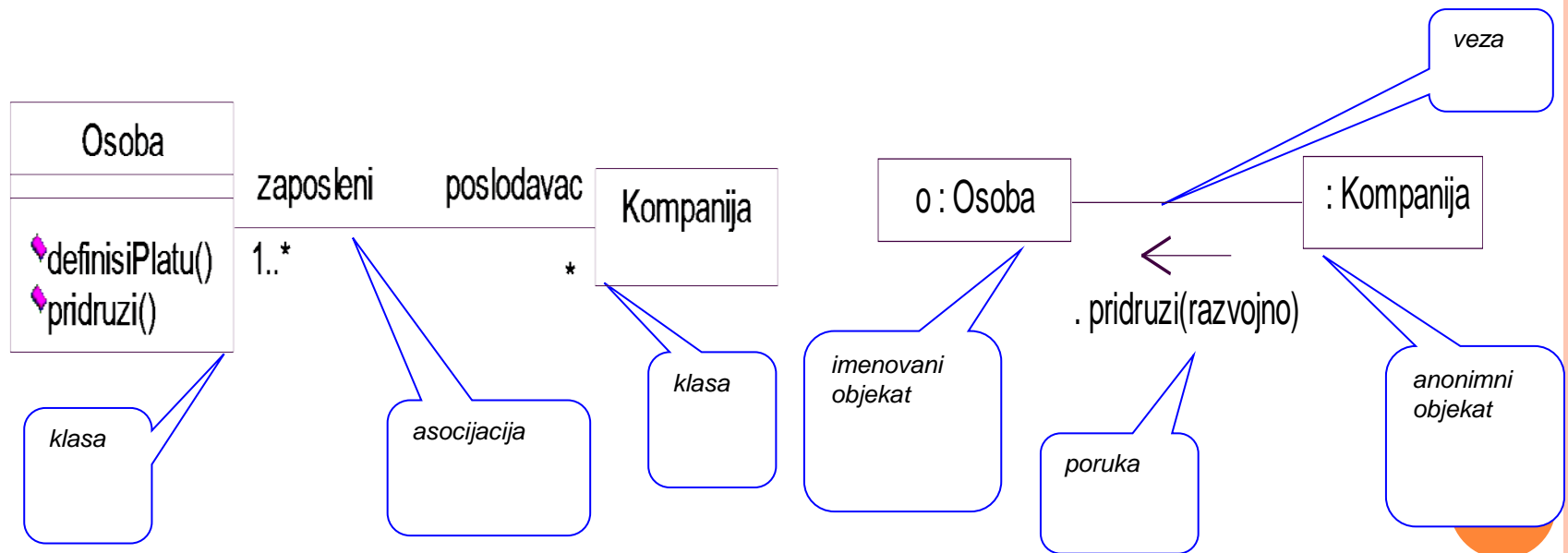
# VEZE (1)

- Veza (*link*) je semantička sprega između objekata
- Veza je instanca relacije asocijacije između odgovarajućih klasa

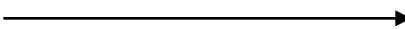
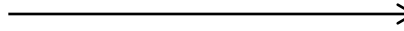
## ○ Primer:

- klasni dijagram (kolaboracije)

dijagram interakcije



## VEZE (2)

- Veza specificira putanju duž koje jedan objekat može da upućuje poruke drugom
- Simbol UML-a za poruku: A solid black arrow pointing to the right.
- Simbol u alatu Rose za poruku: A solid black arrow pointing to the right, with a small open circle at the tail.





## UKRASI VEZA

- Veza se može ukrasiti većinom ukrasa asocijacije (ime, uloge, navigabilnost, agregacija)
- Izuzetak je **multiplikativnost**, jer u vezi uvek učestvuju samo po 1 objekat sa svake strane
- Posebnu kategoriju ukrasa veza čine ukrasi vidljivosti druge strane veze



## UKRASI VIDLJIVOSTI U VEZI

- Specificiraju način na koji objekat koji šalje poruku "vidi" objekat sa druge strane veze
- Tekstualni ukrasi koji se pišu kao stereotipovi na udaljenom kraju veze (kod primaoca)
- Ukrasi:
  - **association** – specificira da je objekat vidljiv jer postoji asocijacija između klasa
  - **self** – specificira da je objekat vidljiv jer sam sebi šalje poruku
  - **global** – specificira da je objekat vidljiv jer je u nekom okružujućem opsegu tj. domenu
  - **local** – specificira da je objekat vidljiv jer je u lokalnom opsegu
  - **parameter** – specificira da je objekat vidljiv jer je argument operacije



## SLANJE I PRIJEM PORUKE

- Prijem jedne poruke se može smatrati instancom jednog događaja
- Kada se pošalje poruka sledi akcija – izvršenje naredbe koja predstavlja apstrakciju metoda-operacija

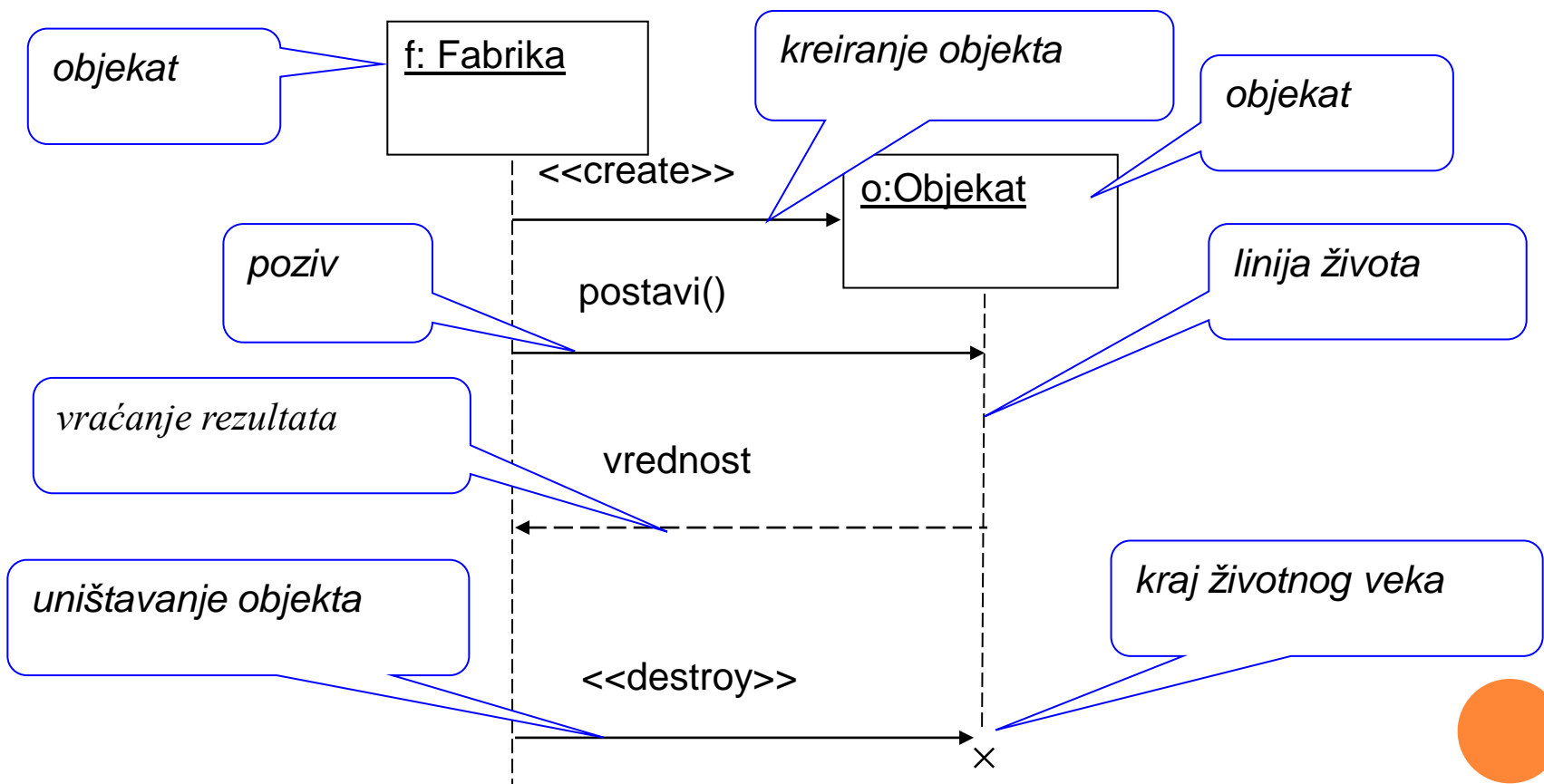
### UML predviđa sledeće vrste akcija poruke-vrednost:

- poziv (*call*) – pokreće operaciju objekta primaoca (može biti i poziv sebi) operacija
- povratak (*return*) – vraća vrednost pozivaocu
- slanje (*send*) – asinhrono se šalje signal primaocu
- kreiranje (*create*) – kreira se objekat <<create>>
- uništavanje (*destroy*) – uništava se objekat (objekat može biti i suicidan) <<destroy>>
- postajanje (*become*) – objekat menja prirodu (na obe strane veze je isti objekat) <<become>>

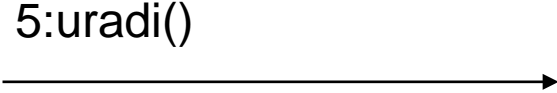


# SLANJE I PRIJEM PORUKE - PRIMER

- Primer dijagrama sekvence i raznih vrsta poruka



## SEKVENCIROVANJE PORUKA (1)

- Unutar svakog toka kontrole neke procesne niti poruke su uređene u vremensku sekvencu
- U dijagramima kolaboracije sekvenca se modelira rednim brojem poruke ispred imena
- Grafička notacija: 
- U dijagramima sekvence sekvenca se modelira implicitno ređanjem poruka odozgo-naniže



## SEKVENCIranJE PORUKA (2)

- Proceduralni (ugnježdeni) tok kontrole se prikazuje strelicama sa popunjenom glavom
- rednih brojevi poruka imaju hijerarhijsku strukturu (nivoi hijerarhije se razdvajaju tačkom)

2.1.3:op() →

- Ravni (*flat*) tok kontrole se prikazuje običnim strelicama

5:op() →

- redni brojevi poruka nemaju hijerarhijsku strukturu

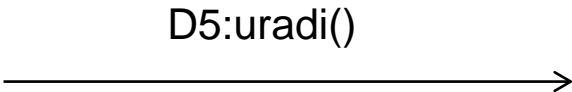


## SEKVENCIranJE PORUKA (3)

- Primer ravne sekvence (Rational Rose koristi tačku umesto dvotačke):



## SEKVENCIJANJE PORUKA (4)

- D5:uradi()
- Identifikacija niti iz koje se izdaje poruka se piše ispred rednog broja poruke u sekvenci
- primer: 
- 5 poruka u toku kontrole koji pripada niti D





## ARGUMENTI I REZULTAT PORUKE

- U okviru poruke se mogu prikazati i njeni argumenti, kao i vraćena vrednost
- primer: `1.2: osoba:=nadji("Petar Petrović")`

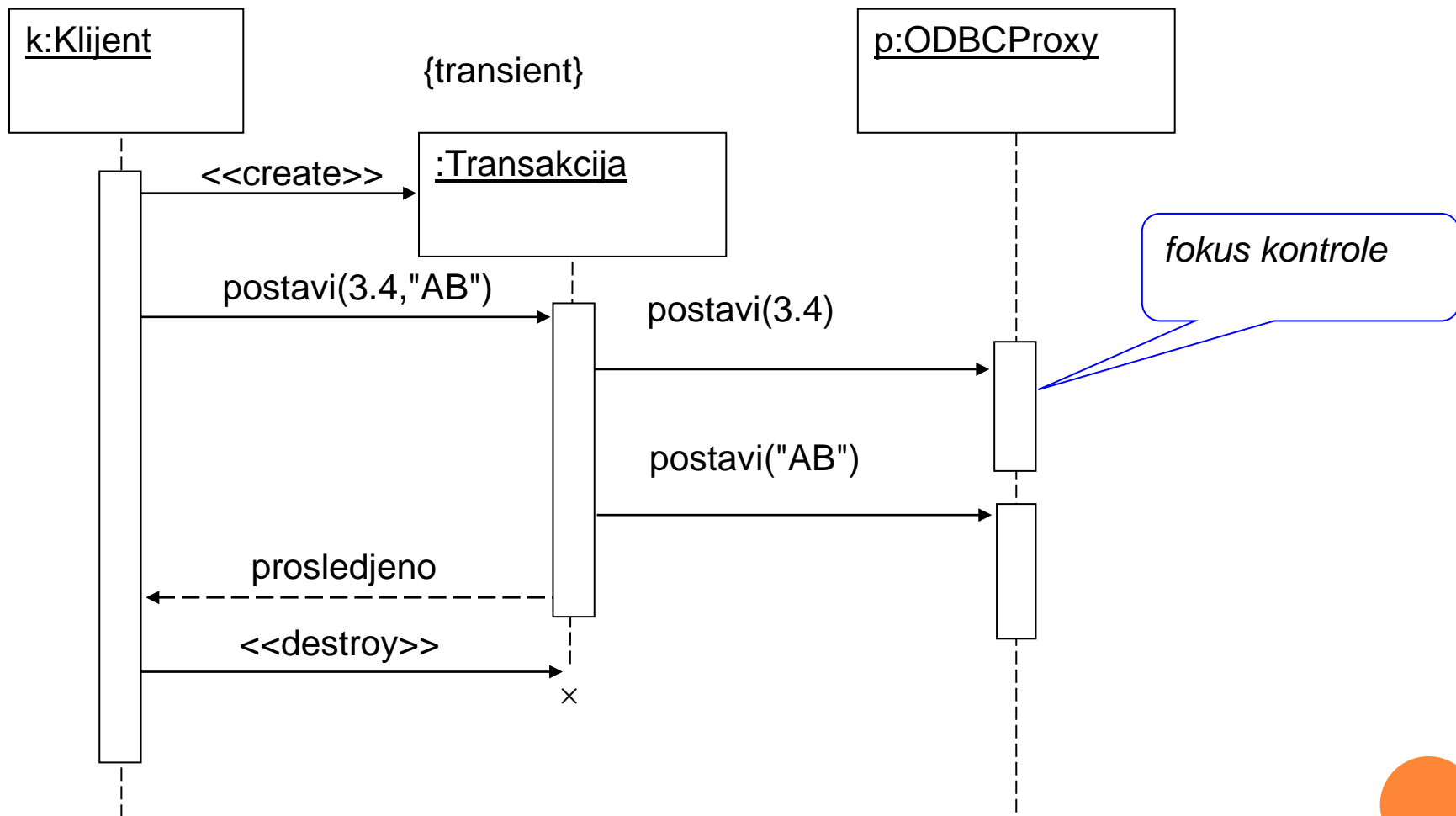


## ŽIVOTNI VEK OBJEKATA I VEZA

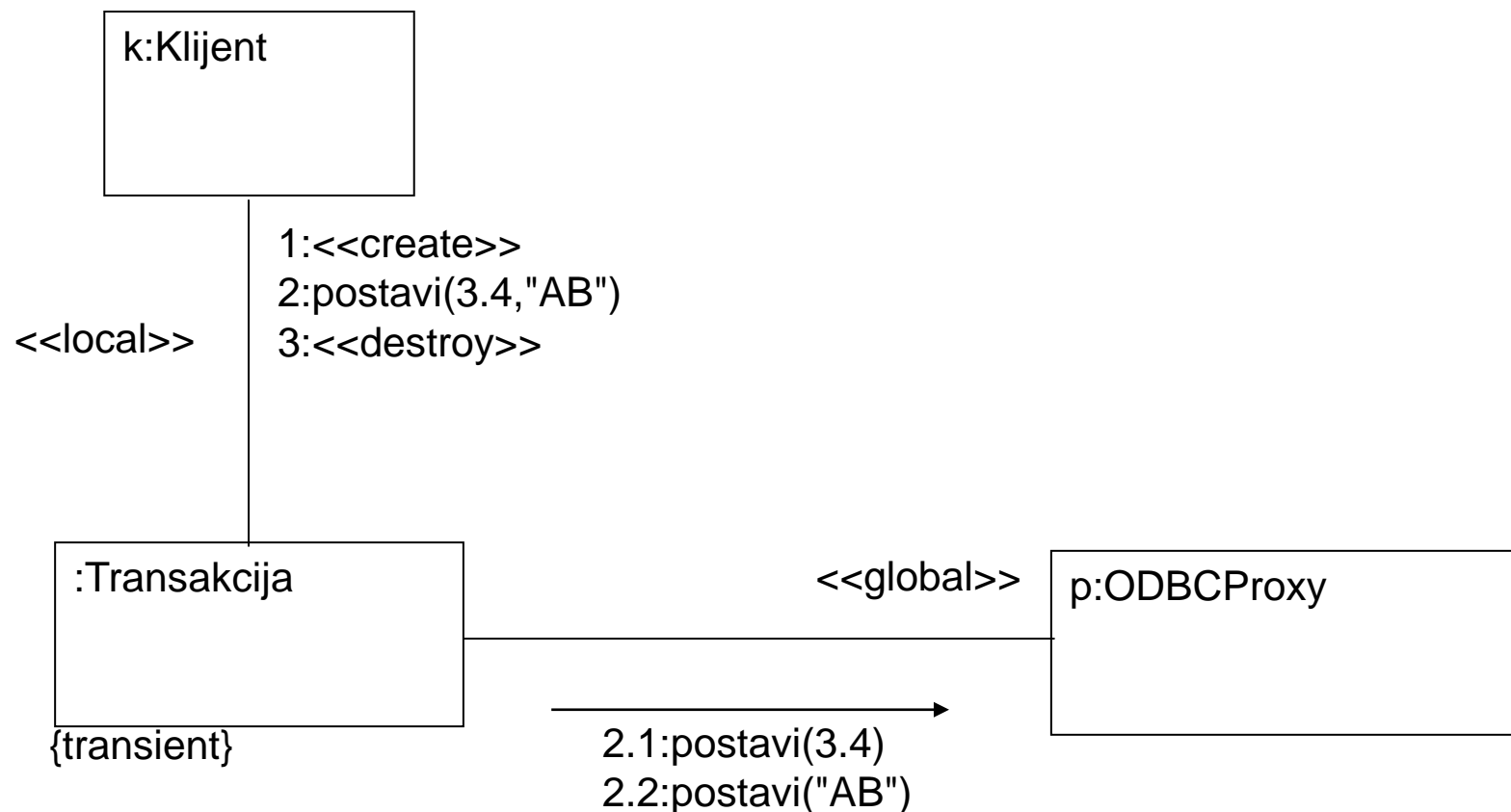
- Ponekad se životni vek objekta ili veze ne poklapa sa trajanjem interakcije
- Objekti i veze mogu nastajati i nestajati u toku interakcije
- Sledeća ograničenja se mogu pripisati objektu i/ili vezi
- `{new}` – objekat/veza se kreira za vreme izvršenja interakcije
- `{destroyed}` – objekat/veza se uništava pre završetka interakcije
- `{transient}` – objekat/veza se kreira i uništava za vreme interakcije
- Promena stanja ili uloge objekta na dijagramu interakcije se naznači njegovom **replikacijom**
- na dijagramu sekvence sve varijante jednog objekta se smeštaju na istu vertikalnu liniju
- na dijagramu kolaboracije varijante se povezuju porukom `<<become>>`



# PRIMER DIJAGRAMA SEKVENCE



# PRIMERI DIJAGRAMA KOLABORACIJE



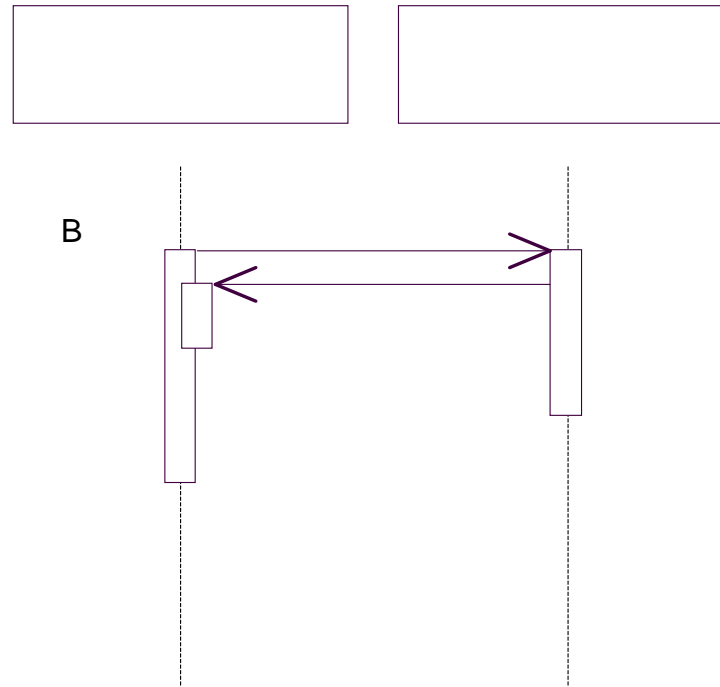
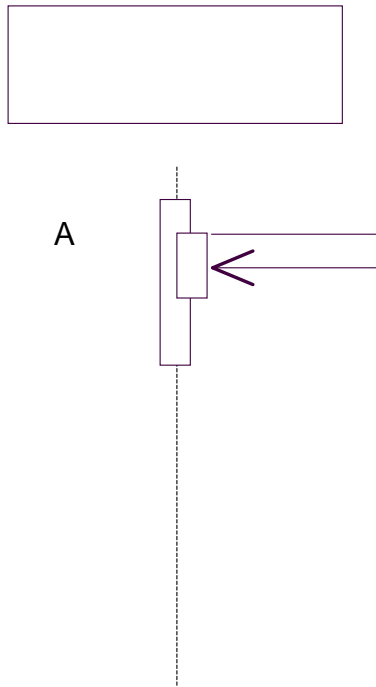
## FOKUS KONTROLE (1)

- Fokus kontrole se može naznačiti samo na dijagramima sekvence
- **Fokus kontrole** definiše period u toku kojeg objekat obavlja jednu akciju direktno ili indirektno kroz podređene operacije
- Moguće je i ugnežđivanje fokusa kontrole iz sledećih razloga:
  - zbog rekurzije ili poziva sopstvene (druge) operacijeAB
  - zbog povratnog poziva (*call back*) od pozvanog objekta




## FOKUS KONTROLE (2)

- Grafička notacija:

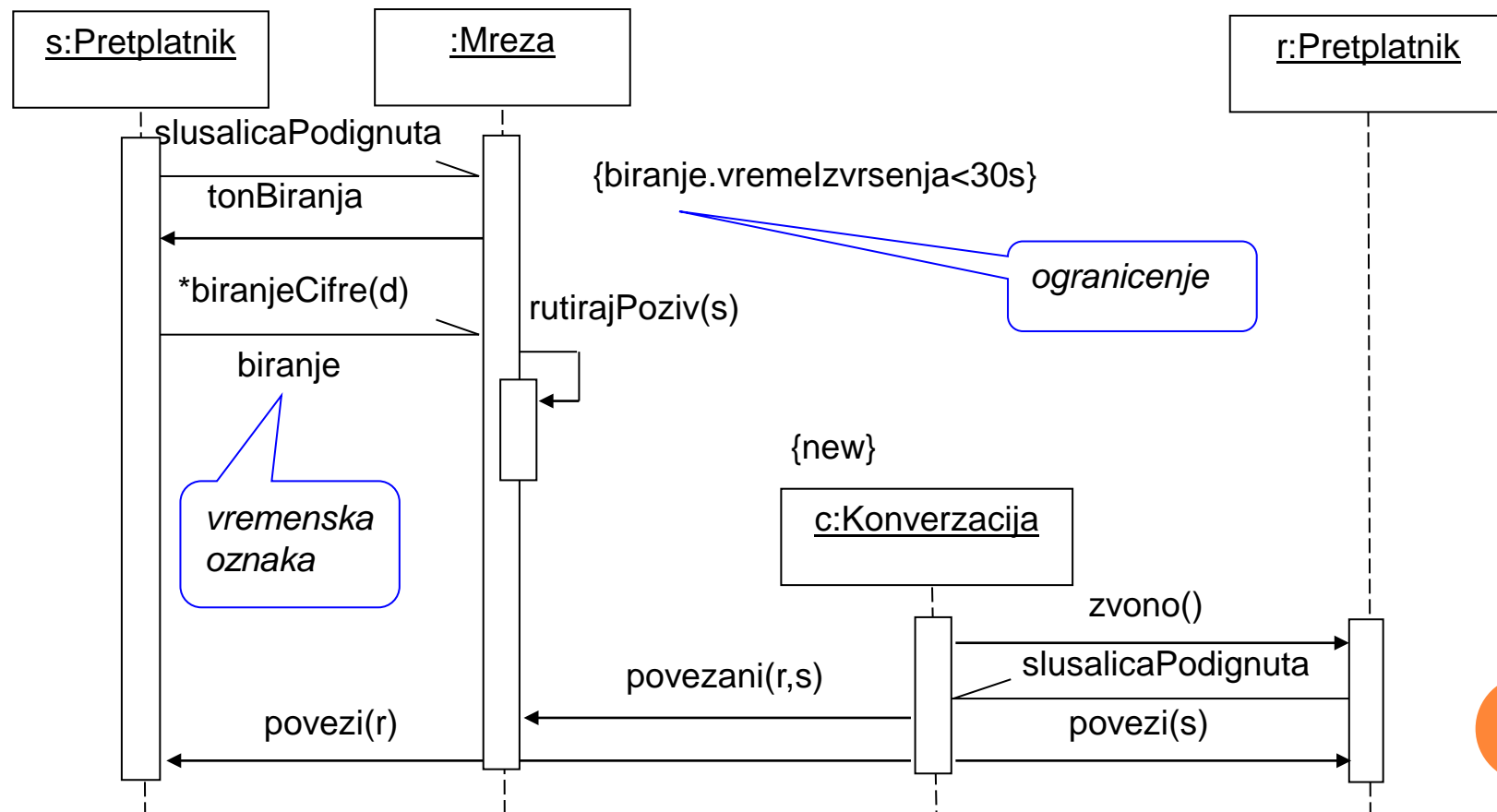


## ITERACIJE I GRANANJE

- Iteracije se modeliraju tako što se iteracioni izraz piše ispred broja poruke:  
 **$*[i:=1..n]$**  ili samo  **$*$**
  - Poruka se ponavlja u skladu sa izrazom
  - Grananje se modelira tako što se uslovni izraz piše ispred broja poruke:  
 **$[x>0]$**
  - Dve ili više poruka u interakciji mogu imati isti redni broj, ali disjunktne uslove
- 

# PRIMER DIJAGRAMA SEKVENCE

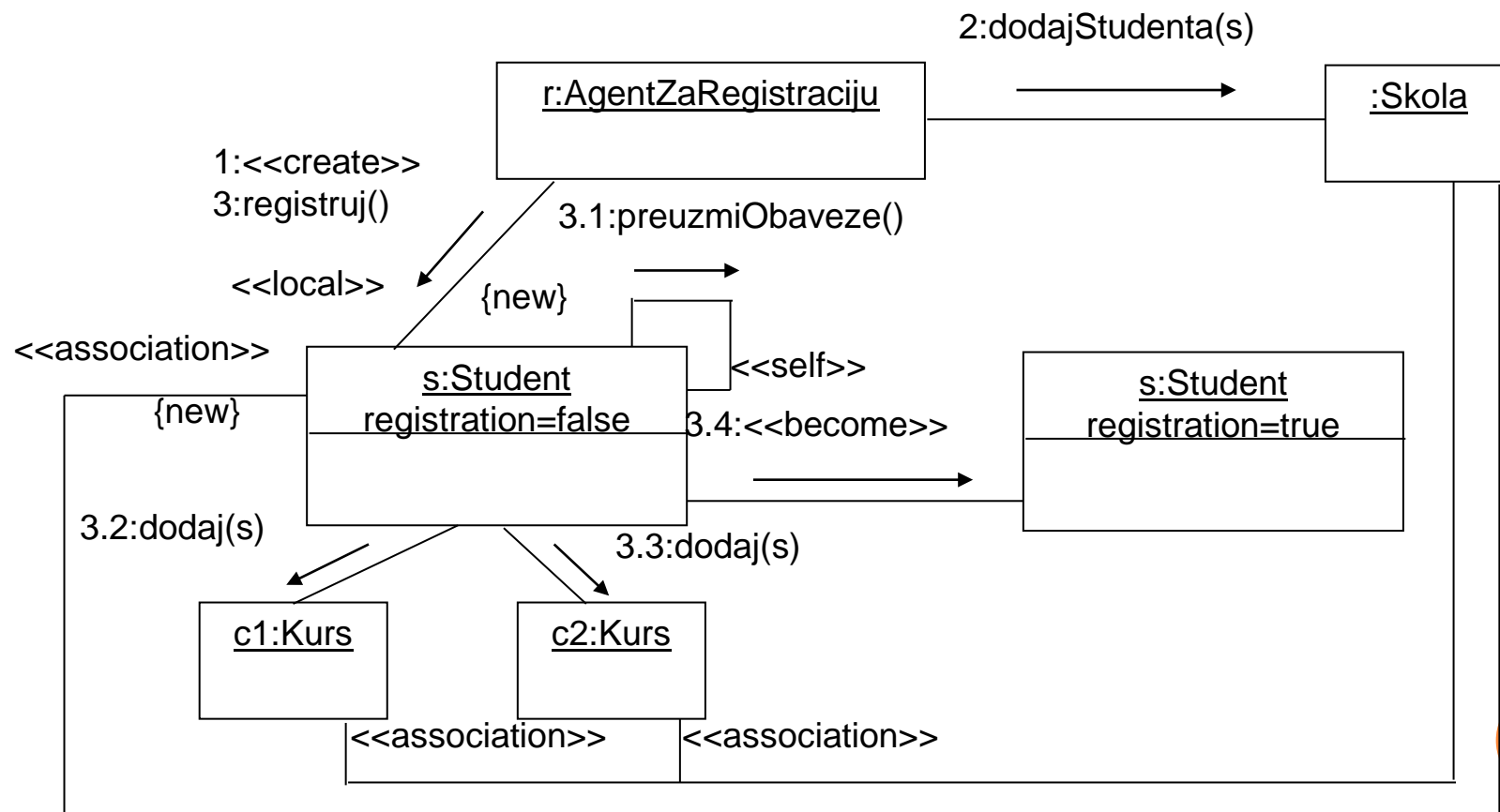
## ○ Uspostavljanje telefonske veze





# PRIMER DIJAGRAMA KOLABORACIJE

## ○ Registracija novog studenta



# MODELIRANJE TOKA KONTROLE POMOĆU VREMENSKOG REDOSLEDA (1)

- Definirati kontekst interakcije; on može biti:
  - ponašanje sistema, podsistema, operacije, klase ili
  - scenario slučaja korišćenja ili kolaboracije
- Identifikovati koji objekti igraju ulogu u interakciji
  - poređati objekte sleva u desno počevši od važnijih objekata



# MODELIRANJE TOKA KONTROLE POMOĆU VREMENSKOG REDOSLEDA (2)

- Definirati linije životnih vekova za svaki objekat
- Za objekte koji se kreiraju i/ili uništavaju za vreme interakcije
  - skratiti linije života
  - eksplicitno naznačiti njihovo rađanje i umiranje sa odgovarajućim stereotipnim porukama



# MODELIRANJE TOKA KONTROLE POMOĆU VREMENSKOG REDOSLEDA (3)

- Počevši od poruke koja inicira interakciju, povlačiti horizontalne linije za poruke između životnih linija objekata, sledeći njihov vremenski redosled odozgo-naniže
- Prikazati osobine poruka kao što su parametri, ako je neophodno da se objasni semantika



# MODELIRANJE TOKA KONTROLE POMOĆU VREMENSKOG REDOSLEDA (4)

- Ako je potrebno vizuelizovati ugnežđivanje poruka ili tačaka u vremenu kada počinju obrade
  - uvesti fokuse kontrole na linije života objekata
- Ako je potrebno specificirati vremenska ili prostorna ograničenja
  - ukrasiti poruke vremenskim oznakama i prikazati odgovarajuća vremenska/prostorna ograničenja
- Ako je potrebno specificirati tok kontrole na formalniji način
  - pridružiti **pred-** i **post-uslove** svakoj poruci



# MODELIRANJE TOKA KONTROLE POMOĆU ORGANIZACIJE (1)

- Definirati kontekst interakcije
- Identifikovati koji objekti igraju ulogu u interakciji
  - poređati objekte kao temena u grafu, stavljajući značajnije objekte u centar dijagrama
- Definirati inicijalne osobine objekata
  - ako se osobine (vrednosti atributa, obeležene vrednosti, stanje ili uloga) značajnije menjaju za vreme interakcije – udvojiti objekat i povezati ga sa originalom <<become>> porukom



# MODELIRANJE TOKA KONTROLE POMOĆU ORGANIZACIJE (2)

- Specificirati veze između objekata preko kojih se mogu razmenjivati poruke
- prvo prikazati veze asocijacije, jer su najvažnije (predstavljaju strukturne konekcije)
- prikazati ostale veze i ukrasiti ih stereotipovima putanja

( <<global>> , <<local>> , ... )



# MODELIRANJE TOKA KONTROLE POMOĆU ORGANIZACIJE (3)

- Počevši od poruke koja inicira interakciju, pridružiti poruke vezama, definišući broj poruke
  - za prikazivanje ugnježđivanja primeniti hijerarhijsku brojnu šemu (notacija sa tačkom)
- Ako je potrebno specificirati vremenska ili prostorna ograničenja
  - ukrasiti poruke vremenskim oznakama i prikačiti odgovarajuća vremenska/prostorna ograničenja
- Ako je potrebno specificirati tok kontrole na formalniji način
  - pridružiti pred- i post-uslove svakoj poruci





# PROCESI I NITI

- Proces je "teški" (*heavyweight*) tok kontrole koji ima vlastiti adresni prostor
- Nit je "laki" (*lightweight*) tok kontrole koji deli zajednički adresni prostor sa drugim nitima
- Jedan proces može sadržati više niti
- Proces je jedinica konkurentnosti u operativnim sistemima: procesi konkurišu za resurse
- Procesima je primeren mehanizam za komunikaciju razmena poruka (*message passing*)
- Nitima je primeren mehanizam za komunikaciju deoba zajedničkih podataka (*data sharing*)



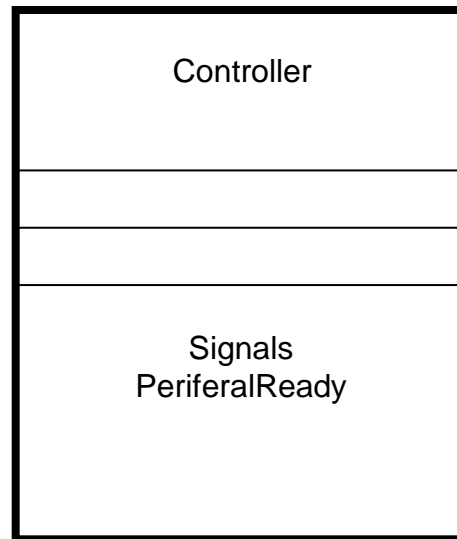
## AKTIVNI OBJEKTI I KLASSE (1)

- Aktivni objekat je onaj koji poseduje vlastiti proces ili nit kontrole
- Aktivna klasa je klasa čije su instance aktivni objekti
- Aktivni objekti su koreni pojedinih tokova kontrole
- Kada se aktivni objekat kreira/uništava – tok kontrole se pokreće/zaustavlja
- Mnogi jezici podržavaju koncept aktivnih objekata (Ada, Java, Smalltalk)
- Aktivni objekti i aktivne klase se praktično mogu pojaviti u dijagramima gde i pasivni



## AKTIVNI OBJEKTI I KLASSE (2)

- Grafički simbol aktivne klase je pravougaonik sa debljom linijom
- Poseban odeljak se predviđa za signale koje klasa može da prima



# SEKVENCIJALNI I KONKURENTNI SISTEMI

- U sekvencijalnim sistemima postoji samo jedan tok kontrole
- U konkurentnim sistemima postoji više uporednih tokova kontrole
- Ako hardverski sistem ima više procesora moguć je stvarni paralelizam tokova kontrole
- Ako postoji samo jedan procesor – izvršavanje tokova je konkurentno
- Konkurentno izvršavanje stvara iluziju paralelnog izvršavanja



# KOMUNIKACIJA

- Pasivni objekat šalje poruku pasivnom: običan poziv operacije
- Aktivan objekat šalje poruku aktivnom – postoji interprocesna komunikacija
  - **randevu**: pošiljalac i primalac se sinhronizuju
  - **poštansko sanduče**: pošiljalac šalje poruku i nastavlja sa radom (asinhrono slanje)
- Aktivan objekat šalje poruku pasivnom
  - nastupaju problemi ako postoji više od jednog aktivnog objekta – potrebna sinhronizacija
- Pasivan objekat šalje poruku aktivnom – ista semantika kao kod dva aktivna objekta



# SINHRONIZACIJA (1)

- Problem nastupa kada je u jednom trenutku više od jednog toka kontrole u objektu
- Ako nije posvećena posebna pažnja – tokovi će interferirati i rezultovati u narušenom stanju
- Problem se naziva i "neizvesnošću trke" (*race hazard*)
- Rešava se uzajamnim isključivanjem (*mutual exclusion*)
- Za uzajamno isključivanje je potrebna sinhronizacija



## SINHRONIZACIJA (2)

- UML predviđa da operacija može biti opisana sinhronizacionom osobinom:
  - sekvencijalna (*sequential*) – pozivaoci se sinhronizuju izvan objekta
  - čuvana (*guarded*) – svi pozivi čuvanih operacija objekta se serijalizuju
  - konkurentna (*concurrent*) – atomska (neprekidiva) operacija
- Sinhronizaciona osobina concurrent može biti modifikovana ograničenjem (*constraint*):
  - npr. da se podrži više čitalaca ali samo jedan pisac

