Programske paradigme i stilovi programiranja

Vladimir Filipović

vladaf@matf.bg.ac.rs

Zahtevi koji se postavljaju pred program

- Da korektno radi, tj. da daje korektne rezultate.
- Da bude lako čitljiv, kako bi se mogao lakše održavati i nadograđivati u budućnosti.

Pregled metodologija programiranja

- "Trik"programiranje – karakteriše tzv. Herojsko doba računarstva. U ovom periodu programi su pisani bez opštih pravila. Korišćenje specifičnosti u radu računara, trikovi za uštedu memorije, pisanje samomodifikujućih programa su bile karakteristika vrhunskog profesionalca. Ali takav pristup je dovodio do kreiranja teško čitljivih programa.
- Strukturno programiranje.
- Objektno orijentisano programiranje

Strukturno programiranje

- Autori prvih radova o strukturnom programiranju su E.W. Dijkstra, A.P. Hoare, N. Wirth i dr.
- Popularnosti strukturnog programiranja doprinela i pojava programskog jezika PASCAL.
- Strukturno programiranje bilo povezano sa automatskim dokazivanjem korektnosti programa.
- Pod strukturnim programiranjem podrazumeva se opšta metodologija za pisanje kvalitetnih programa. Ona sadrži:
  - Programiranje od opšteg ka posebnom (top-down)
  - Modularnost
  - Korišćenje ograničenog broja upravljačkih struktura u zapisu programa

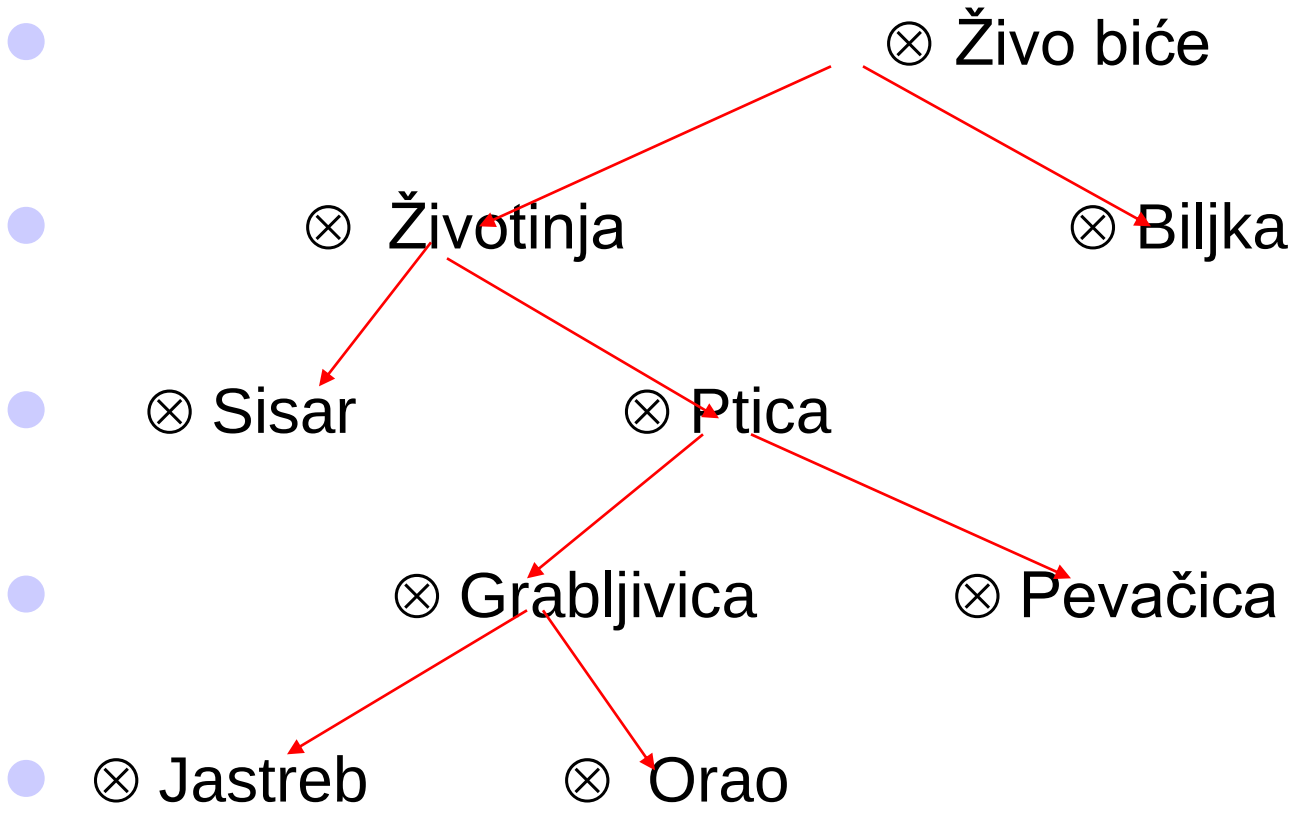
Objektno-orijentisano programiranje

- **Objekat** - integralna celina podataka i procedura za rad sa njima. Zbog prisustva procedura u objektima, objekti imaju mogućnost da samostalno deluju, tj. postaju dinamički.
- **Objektno-orijentisano programiranje** - programska paradigma zasnovana na skupu objekata koji dejstvuju međusobno. Glavne obrade zasnivaju se na manipulisanju objektima.
- **Metod** - funkcija koja je sastavni deo objekta, tj. postupak kojim se realizuje poruka upućena objektu.
- **Poruka** - skup informacija koji se šalje objektu. Sastoji se iz adrese (objekta primaoca poruke) i saopštenja (kazuje šta treba da se uradi).

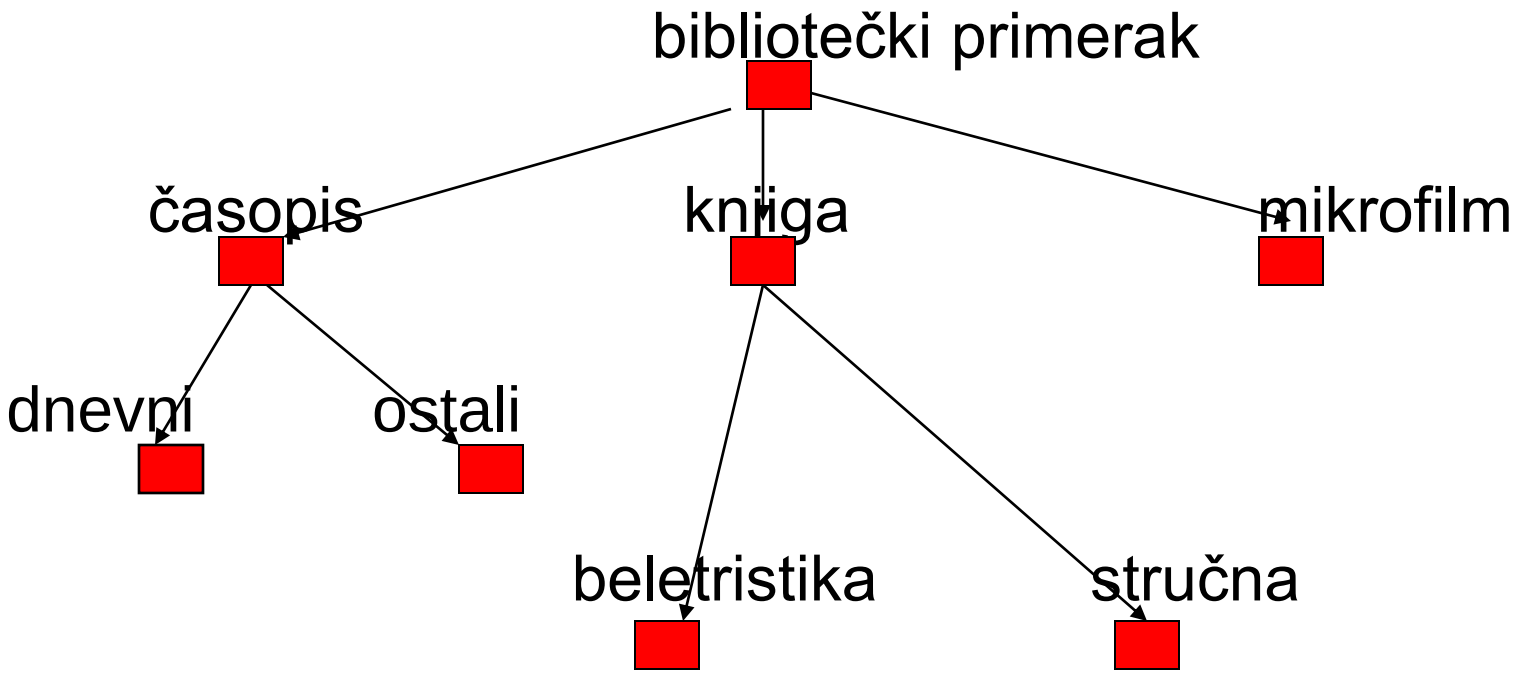
- **Klasa** - Skup objekata sa zajedničkim svojstvima (koji se ponašaju na isti način). Definiše šablon za kreiranje instanci.
- **Primerak (instanca)** - konkretan objekat iz klase. Skup instanci sa skupom metoda kreira jednu klasu.
- Klasa B je **potklasa** klase A ako su svi primerci klase B istovremeno i primerci klase A. Za klasu kažemo da je **nadklasa** klase B. Potklase nastaju dodavanjem novih svojstava (metoda) postojećoj klasi.
- **Nasleđivanje** - mehanizam za kreiranje novih klasa iz postojećih. Nasleđivanjem se formiraju relacije između jedne i više drugih klasa.
- **Polimorfizam** - Mogućnost primene istog metoda (operatora) na primerke različitih klasa.

Primeri nasleđivanja

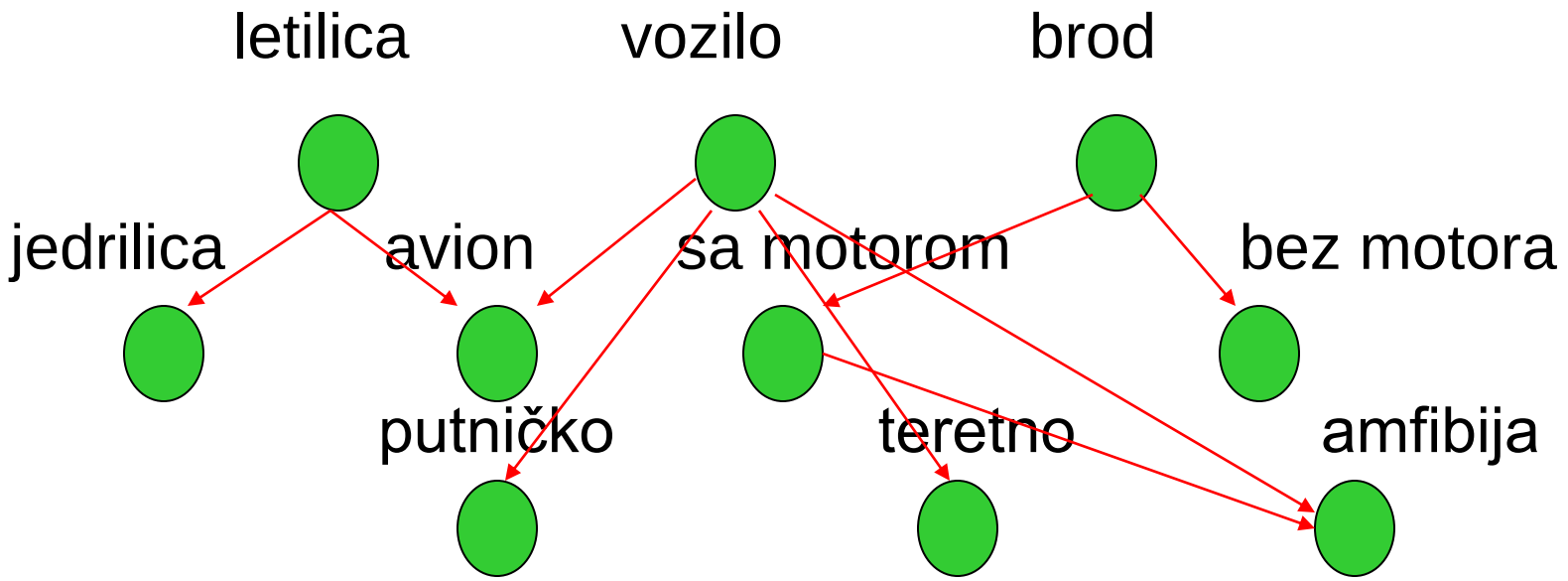
Prilikom projektovanja programa uočavaju se veze između pojedinih klasa i način povezivanja tih klasa sa drugim klasama. Ovde je bitna uloga nasleđivanja.



Još jedan primer hijerarhijskog nasleđivanja



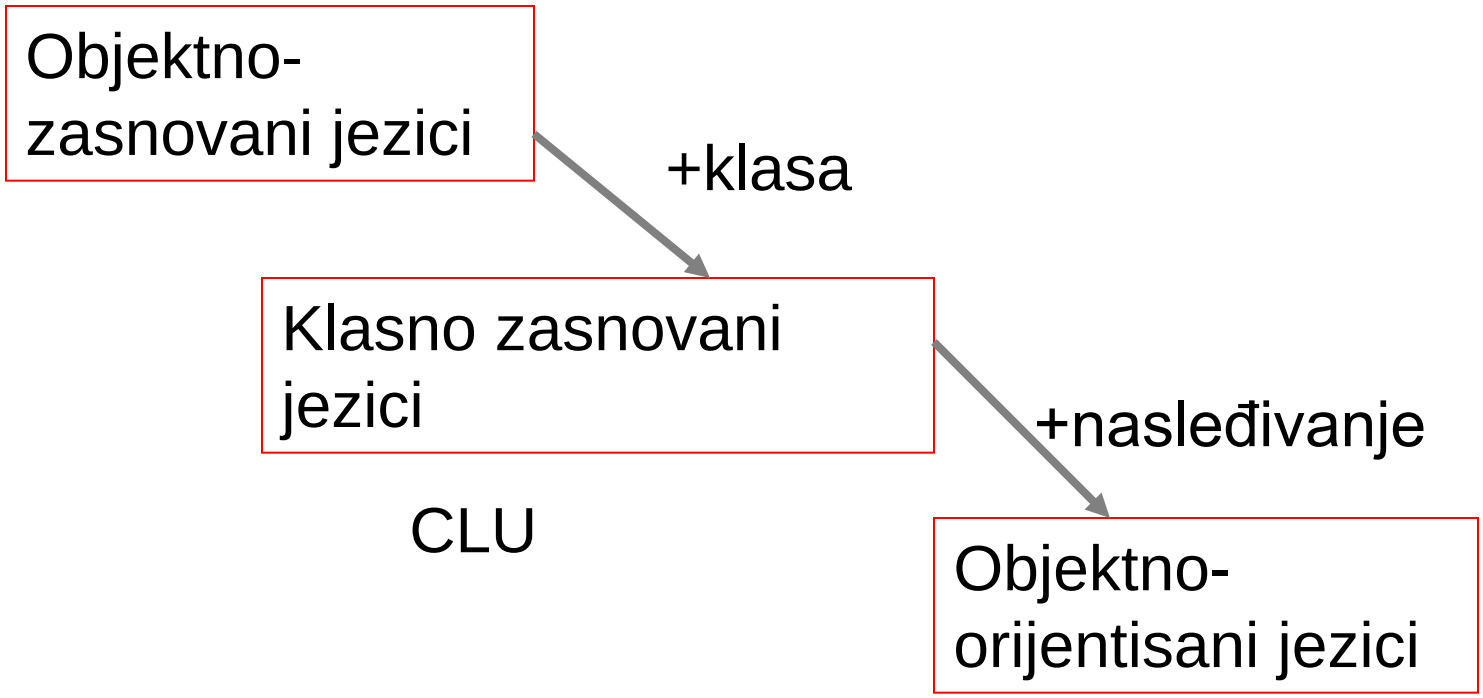
Primer višestrukog nasleđivanja (klasa može imati više direktnih natklasa)



Višestruko nasleđivanje nije podržano u jeziku Java, ali jeste u jeziku C++.

Uloga pojedinih pojmova (Wagner)

Ada, Actors



Smalltalk, Simula 67

Zašto je objektno-orijentisan koncept doživeo veliki uspeh?

- Ponovno korišćenje softvera.
- Pogodan za: analizu, projektovanje i programiranje.
- Najpogodniji za simuliranje događaja.
- Objašnjenje se delimično može naći kroz pogled istorijskog razvoja OOP.
  - 1967: Dall i Simula 67.
  - 70-tih godina: A. Kay i Smalltalk.
  - Razvoj ostalih OO jezika.

Dalji razvoj paradigmi programiranja

- Programiranje vođeno događajima
- Aspekt-orjentisano programiranje
- Ekstremno programiranje
- Obrasci dizajna
- Unifikovani jezik za modeliranje – UML