Projektovanje softvera

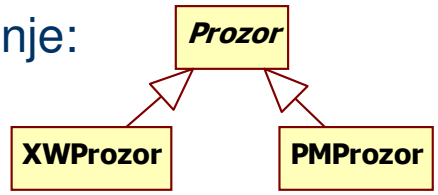
Most

Most (1)

- Ime i klasifikacija:
  - *Most* (eng. *Bridge*) – objektni uzorak strukture
- Namena:
  - razdvaja apstrakciju od njene implementacije da bi mogle da se nezavisno menjaju
- Drugo ime:
  - Ručka/Telo (Handle/Body)

Most (2)

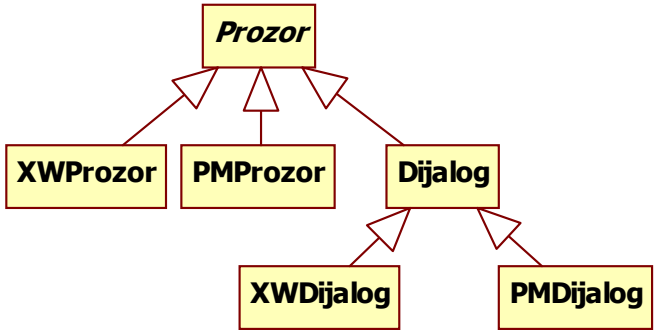
- Motivacija:
  - problem: postoji više implementacija za jednu apstrakciju
  - tradicionalno OO rešenje: apstraktna klasa i izvedene klase
  - primer: apstrakcija `Prozor` u alatima za GUI
    - potrebno je razvijati aplikacije za razne prozorske sisteme
      - npr. *X Window System* i *IBM Presentation Manager*
    - tradicionalno OO rešenje:



  - rešenje nije dovoljno fleksibilno
    - nasleđivanje čvrsto vezuje implementaciju za apstrakciju
    - teško se nezavisno menjaju, proširuju i ponovo koriste
    - generalizacija je statička relacija (ostvarena u vreme prevođenja)
      - implementacija ne može da se promeni dinamički

Most (3)

- Motivacija (nastavak): dva problema
  - (1) uvodi se nova apstrakcija - prozor za dijalog `Dijalog`
    - da bi se apstrakcija implementirala na obe platforme dobija se:



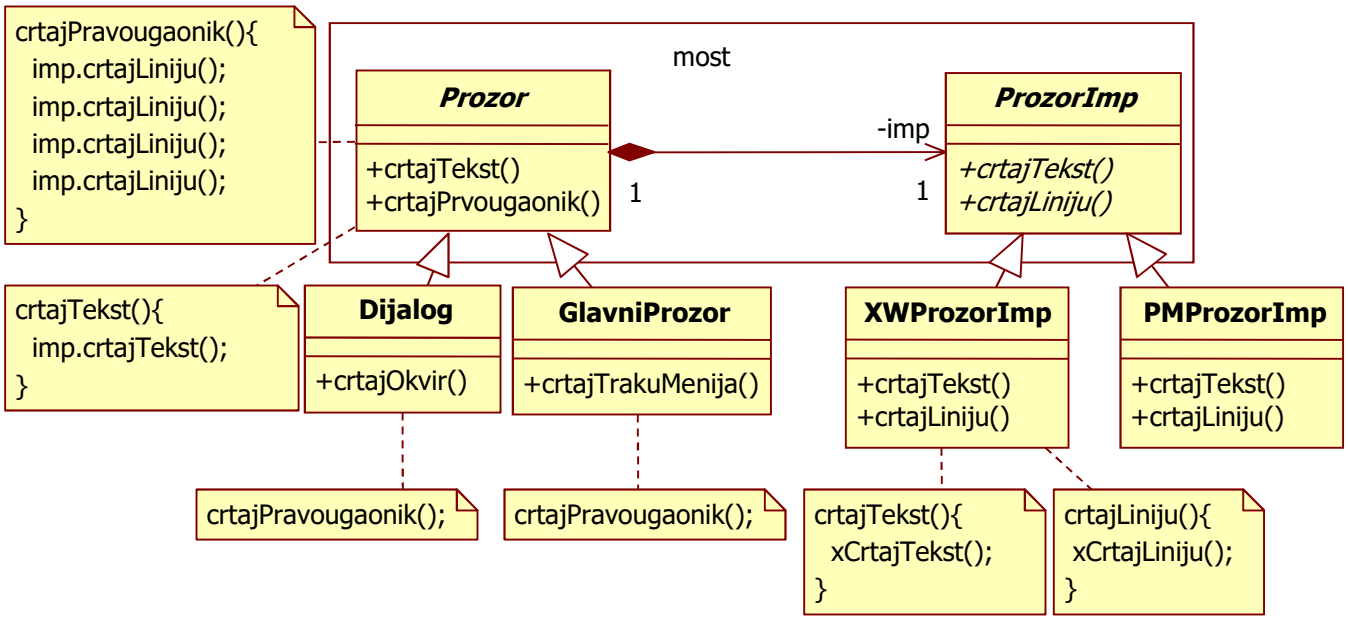
    - za svaku novu vrstu prozora – moraju da se definišu po 2 nove klase implem.
    - podrška za novu platformu – po jedna nova klasa za svaku vrstu prozora

Most (4)

- Motivacija (nastavak):
  - (2) klijentski kod postaje zavisan od platforme
    - kad god klijent pravi prozor – pravi primerak konkretne klase sa specifičnom implementacijom za datu platformu
    - otežano prenošenje klijentskog koda na druge platforme
    - klijenti ne bi trebalo da se vezuju za konkretne implementacije
      - oni treba da vode računa samo o različitim apstrakcijama prozora
      - samo bi implementacija prozora smela da zavisi od platforme
  - rešenje problema: uzorak *Most*
    - apstrakcija prozora i njegova implementacija su dva odvojena korena odgovarajućih hijerarhija klasa
    - uspostavlja se most (asocijacija) između apstrakcije i implementacije
    - konkretne apstrakcije (vrste prozora) i implementacije (platforme) mogu nezavisno da se menjaju

Most (5)

- Motivacija (nastavak):



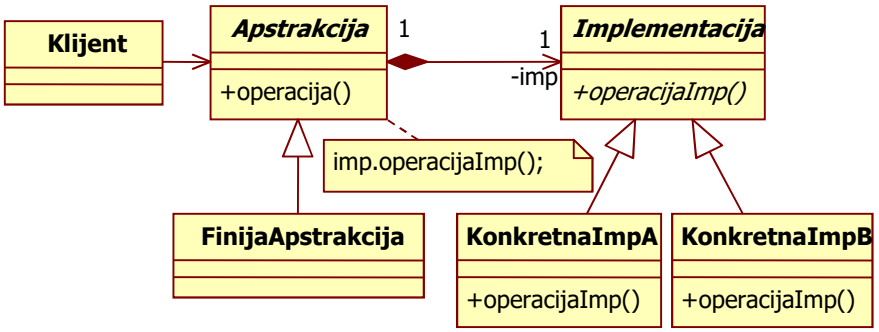
  - sve operacije klase `Prozor` se implementiraju primenom apstraktnih operacija klase `ProzorImp`

Most (6)

- Primenljivost: uzorak treba da se koristi kada
  - treba da se izbegne trajno vezivanje apstrakcije i njene implementacije
    - npr., ako je potrebno da se implementacija menja u vreme izvršenja
  - i apstrakciji i implementaciji je potrebno proširivanje kroz potklase
    - most omogućava kombinovanje različitih apstrakcija i implementacija
  - promena u implementaciji apstrakcije ne sme da utiče na klijente
  - potpuno sakrivanje implementacije klase od klijenta
    - bitno u jeziku kakav je C++
    - definicije klasa su u h fajlovima – delimično se otkriva implementacija
  - postoji opasnost od prevelikog broja klasa
    - npr. u primeru se vidi kvadratni rast (broj sistema x broj vrsta prozora)
  - kada se želi da istu implementaciju deli više objekata
    a da to bude sakriveno od klijenta (eventualno uz brojanje referenci)

Most (7)

- Struktura:



- Saradnja:
  - `Klijent` koristi ugovor *`Apstrakcija`* za pristup objektima klasa `FinijaApstrakcija`
  - `FinijaApstrakcija` prosleđuje klijentske zahteve objektu implementacije preko ugovora *`Implementacija`*

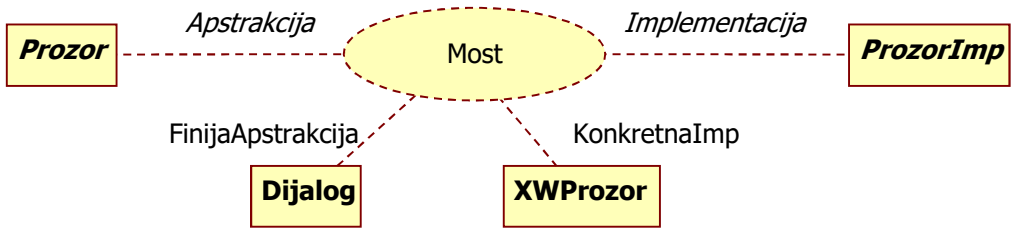
- Učesnici:
  - *`Apstrakcija`* (klasa *`Prozor`*)
    - definiše ugovor apstrakcije
    - održava referencu na objekat implementacije
  - `FinijaApstr.` (klasa `Dijalog`)
    - impementira ugovor apstrakcije
  - *`Implementacija`* (klasa *`ProzorImp`*)
    - definiše ugovor klasa implementacije
      - taj ugovor ne mora da liči na ugovor apstrakcije
  - `KonkretnaImpX` (klase `XWProzorImp`, `PMProzorImp`)
    - implementira ugovor implementacije
    - definiše konkretnu implementaciju

Most (8)

- Posledice:
  - razdvajanje implementacije od ugovora – nema trajnog vezivanja
    - implementacija može da se konfiguriše dinamički
    - eliminisane su zavisnosti implementacije i apstrakcije u vreme prevođenja
      - izmena klasa apstrakcije ne zahteva prevođenje klasa implementacije
      - izmena konkretnih implementacija ne izaziva prevođenje klasa apstrakcije
  - bolje mogućnosti proširivanja
    - hijerarhije apstrakcija i implementacija mogu nezavisno da se proširuju
  - skrivanje detalja implementacije od klijenta
    - klijent ne vidi ništa od implementacije, vidi samo apstrakciju
    - bitno kada mora da se obezbedi kompatibilnost verzija biblioteke klasa
      - može da se promeni kompletna implementacija
      - samo ako se menja njen interfejs, menja se apstrakcija
      - klijent se ne menja do god se ne promeni interfejs apstrakcije

Most (9)

- UML notacija:



- Povezani uzorci:
  - *Apstraktna fabrika* može da stvara konkretne Implementacije za više *Mostova*
  - *Adapter* i *Most* prilagođavaju klijentu interfejs neke implementacije
    - *Adapter* se obično projektuje retroaktivno
    - *Most* se obično projektuje unapred