Objektno-orijentisano projektovanje

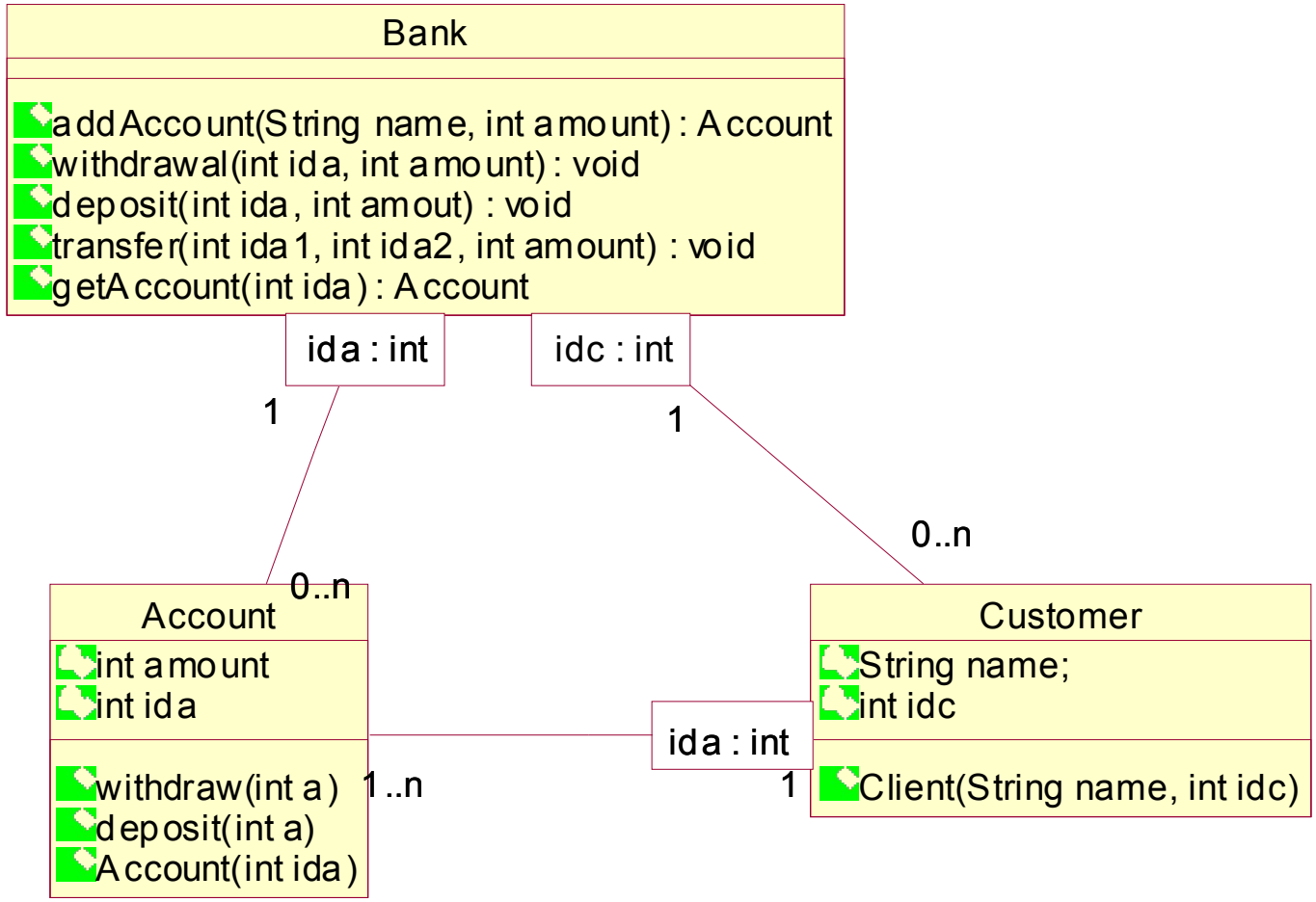
Elektronski fakultet Niš

Projektni obrasci - nastavak

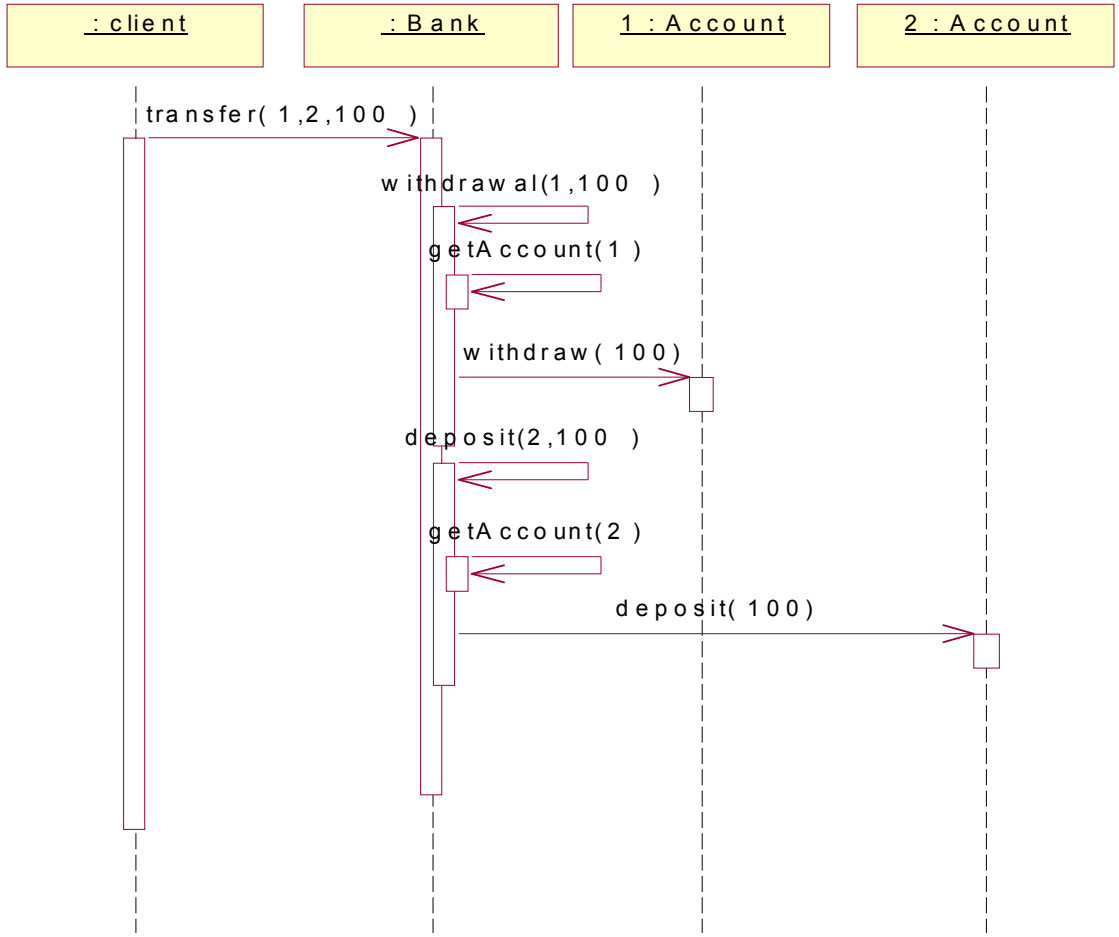
Drugi primer - Banka

- Osnovni bankarski sistem:
  - 1 banka, n računa.
  - Svaki račun pripada jednom klijentu.
  - Svaki račun može biti kreditiran određenom količinom novca.
- Funkcije banke
  - Povlačenje novca sa računa, stavljanje novca na račun, transfer novca sa jednog računa na drugi…

Naivno rešenje



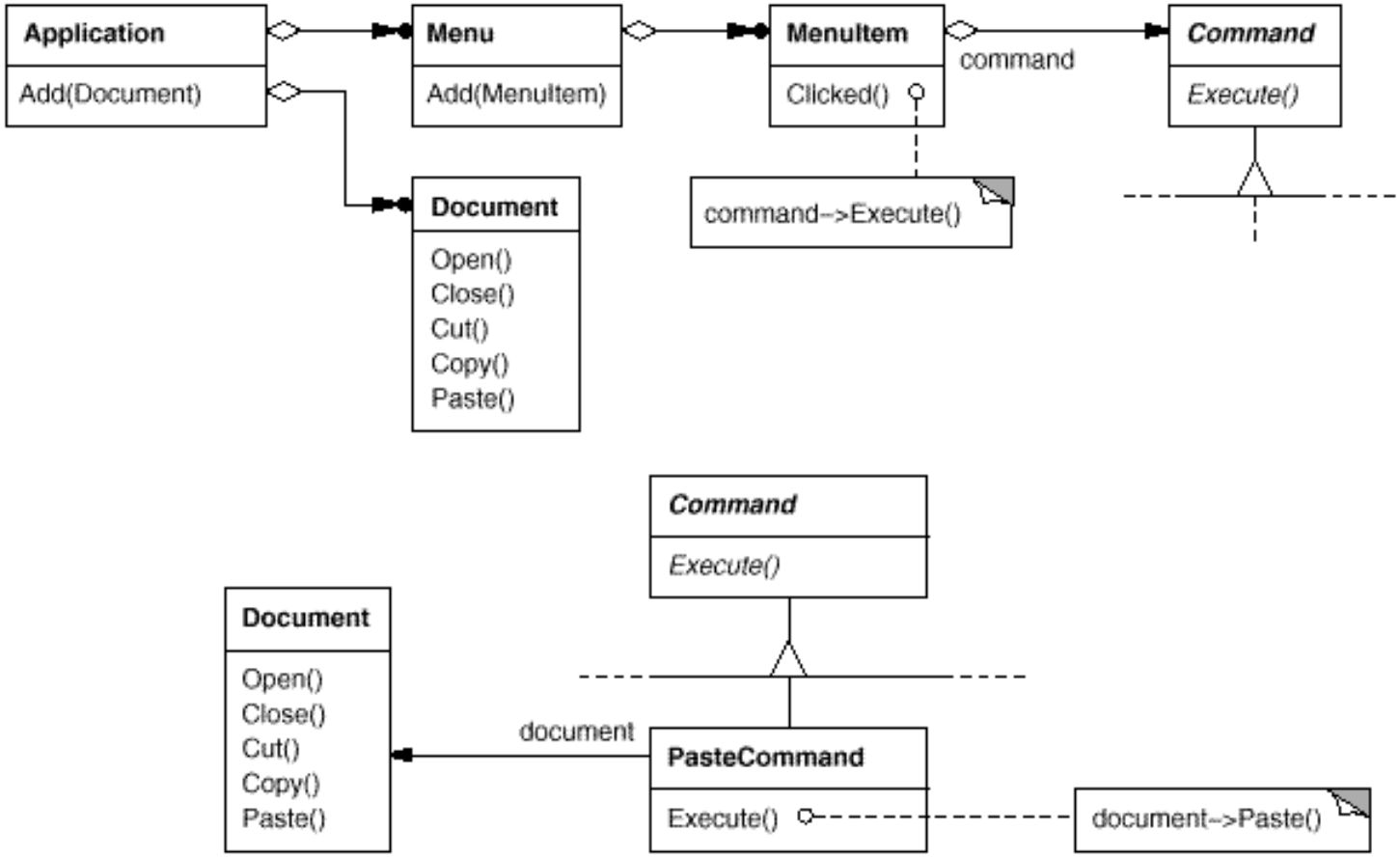
Naivno rešenje



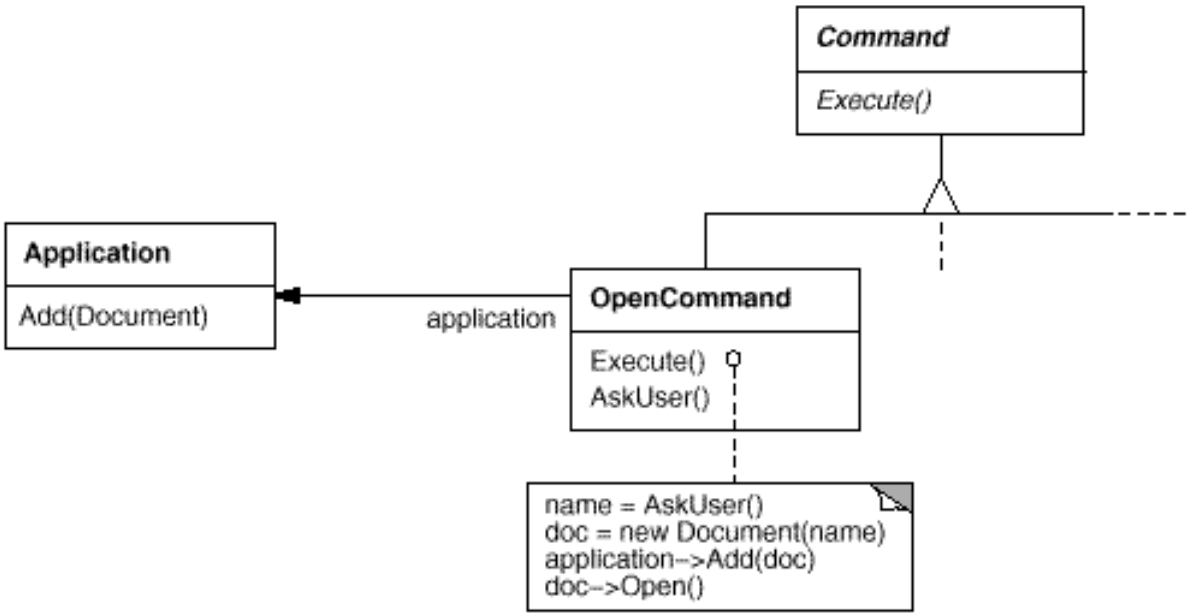
Primena Command Pattern-a

- Enkapsulira zahteve u objekte, i na taj način omogućava parametrizovanje klijenata sa različitim zahtevima, organizovanje reda zahteva, logovanje zahteva, omogućava realizaciju UNDO operacije,....

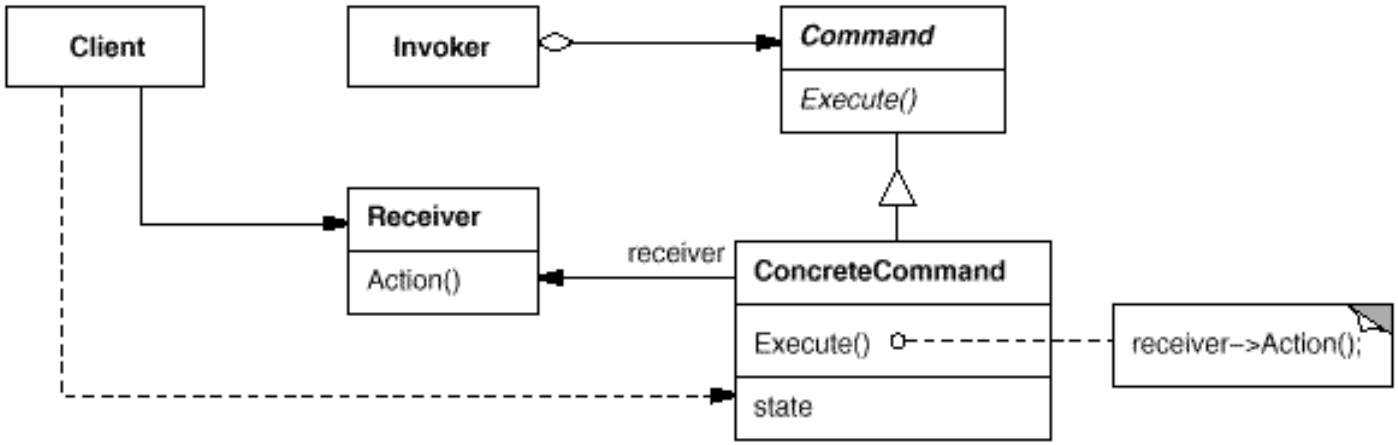
Primer Command obrasca



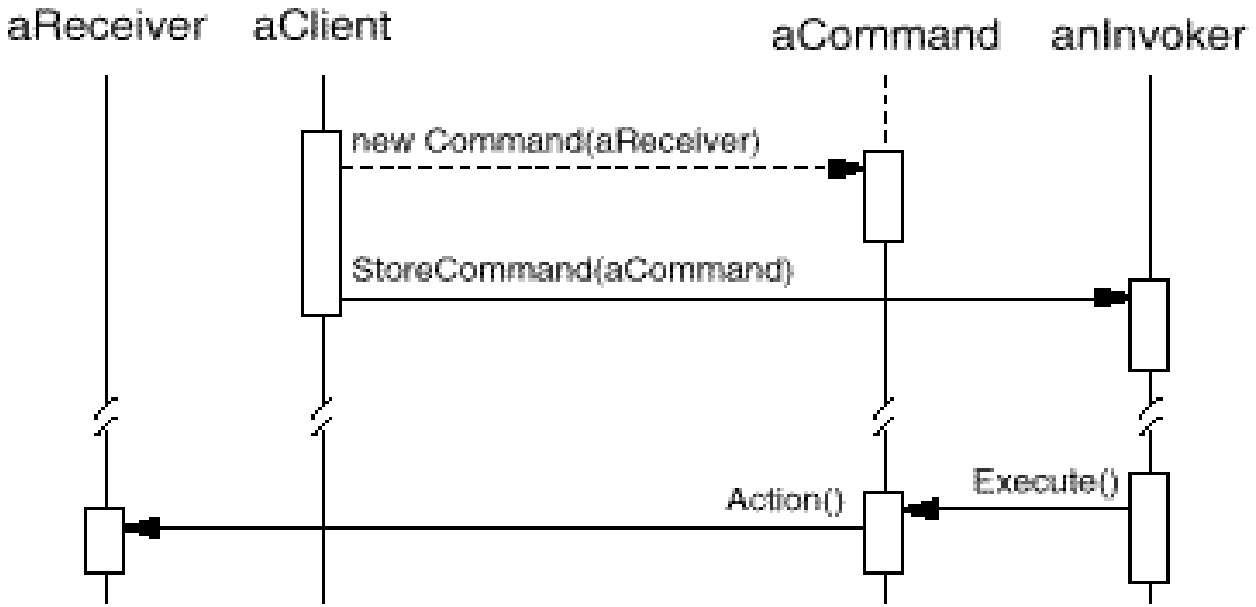
Primer Command obrasca



Struktura Command obrasca



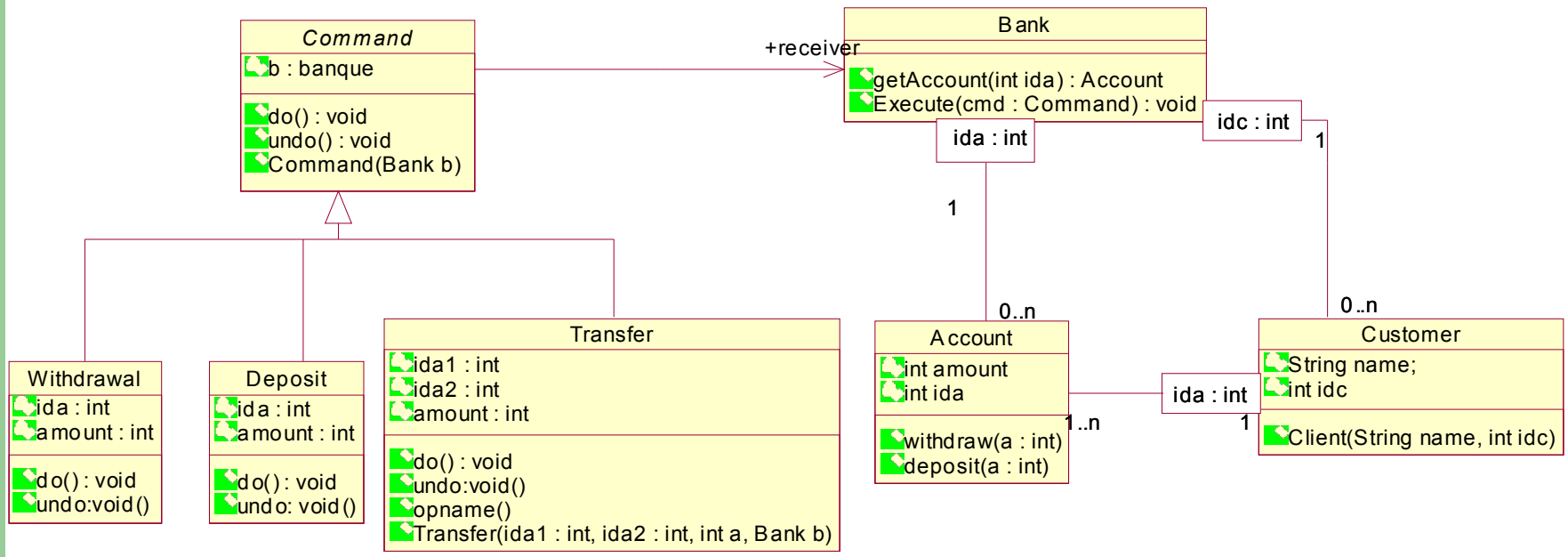
Struktura Command obrasca



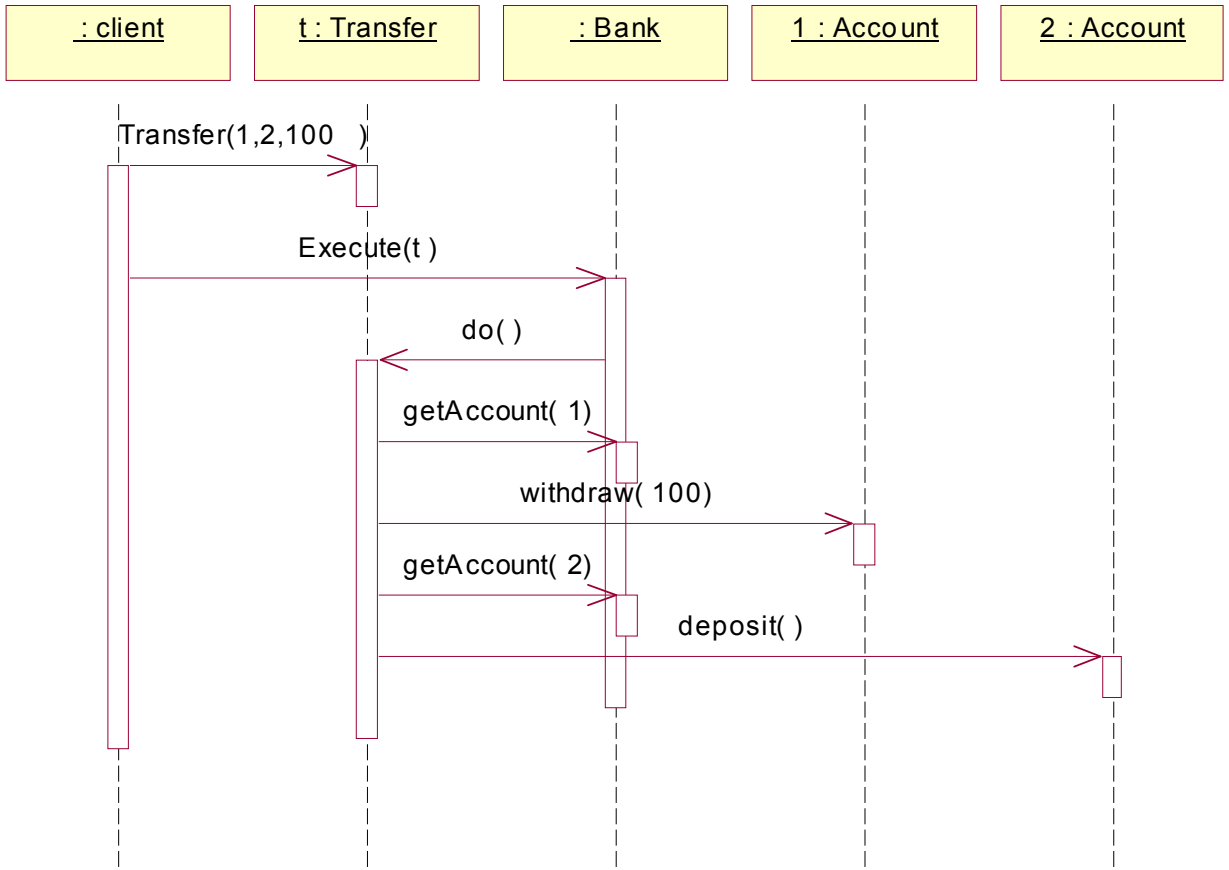
Koristi od primene

- Razdvaja objekte koji pozivaju operaciju od objekata koji znaju kako da izvrše operaciju.
- Komande su objekti. Mogu da se prošire kao i bilo koji drugi objekti.
- Lako je dodavanje novih komandi, jer ne moraju da se menjaju postojeće klase.

Primena Command obrasca



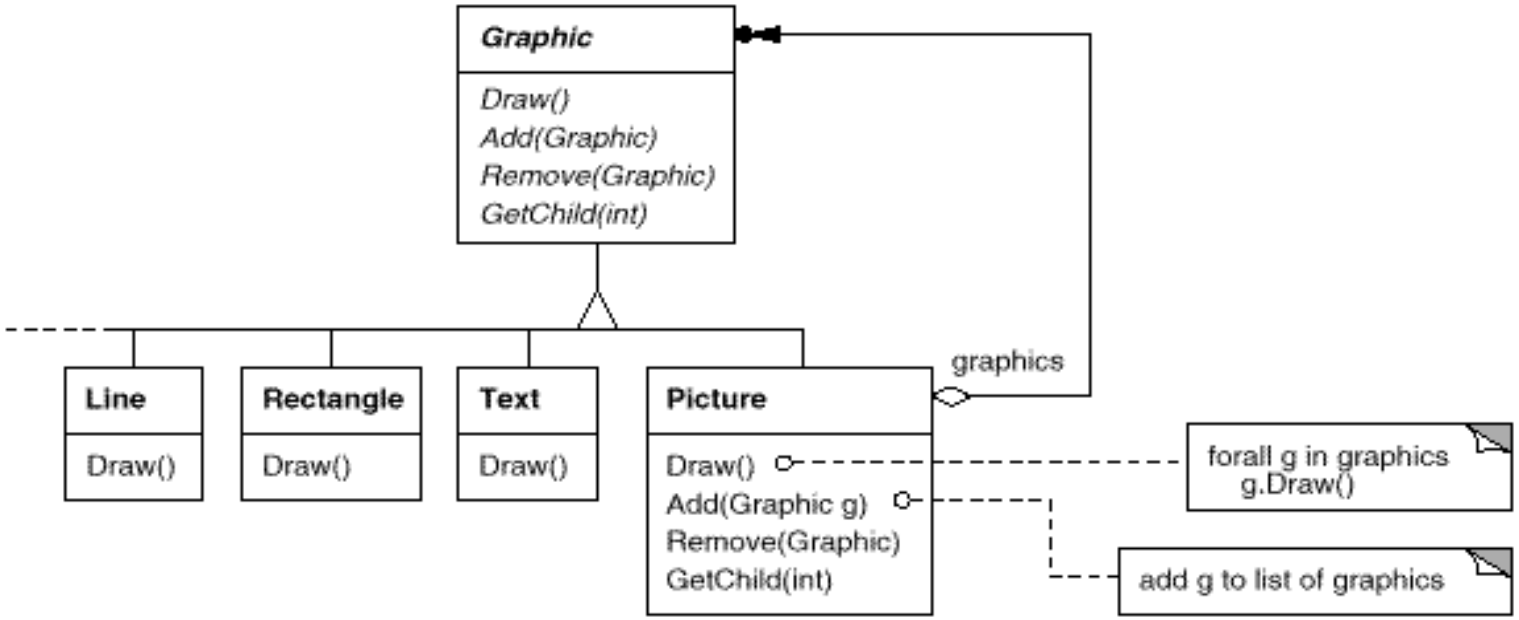
Primena Command obrasca



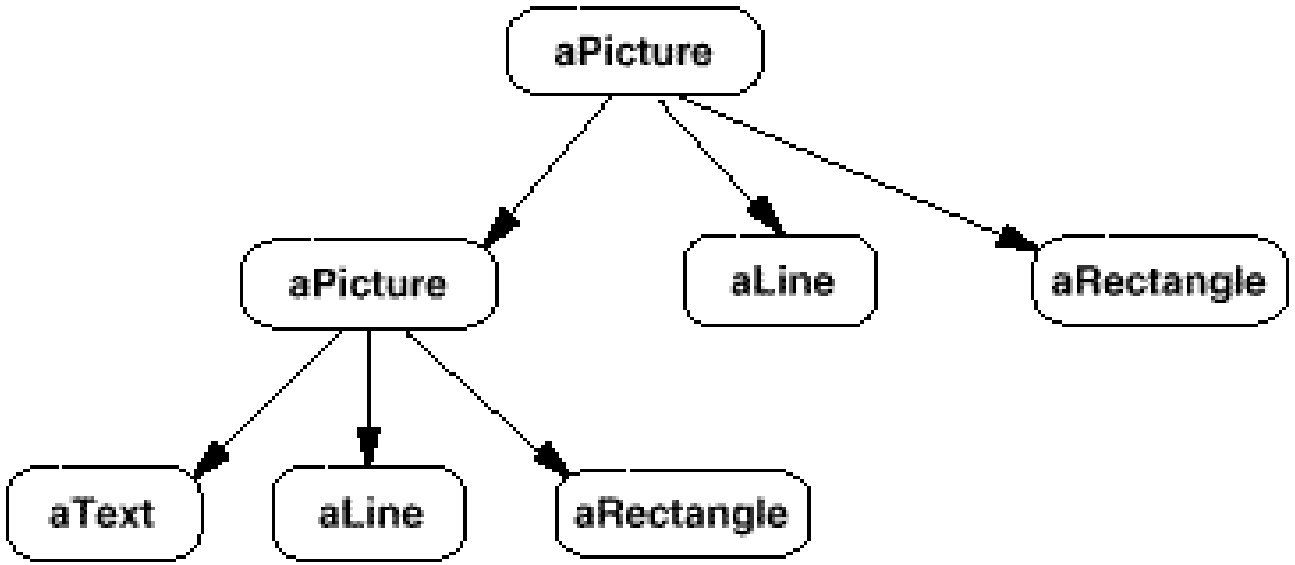
Obrazac Composite

- Organizuje objekte u strukturu stabla kako bi predstavio hijerarhiju **celina-deo** tipa.
- Ovaj obrazac omogućava uniformno tretiranje individualnih objekata i kompozitnih objekata.

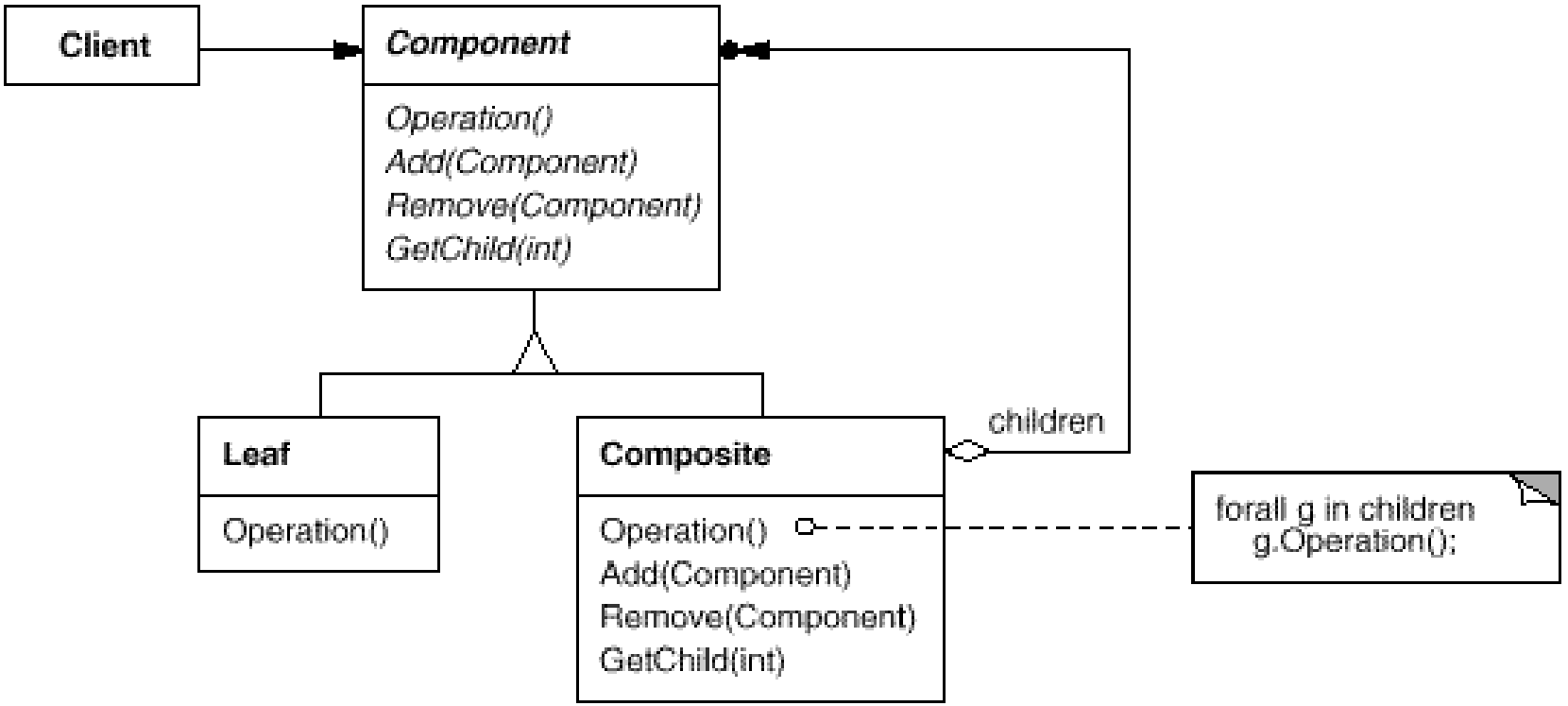
Primer Composite obrasca



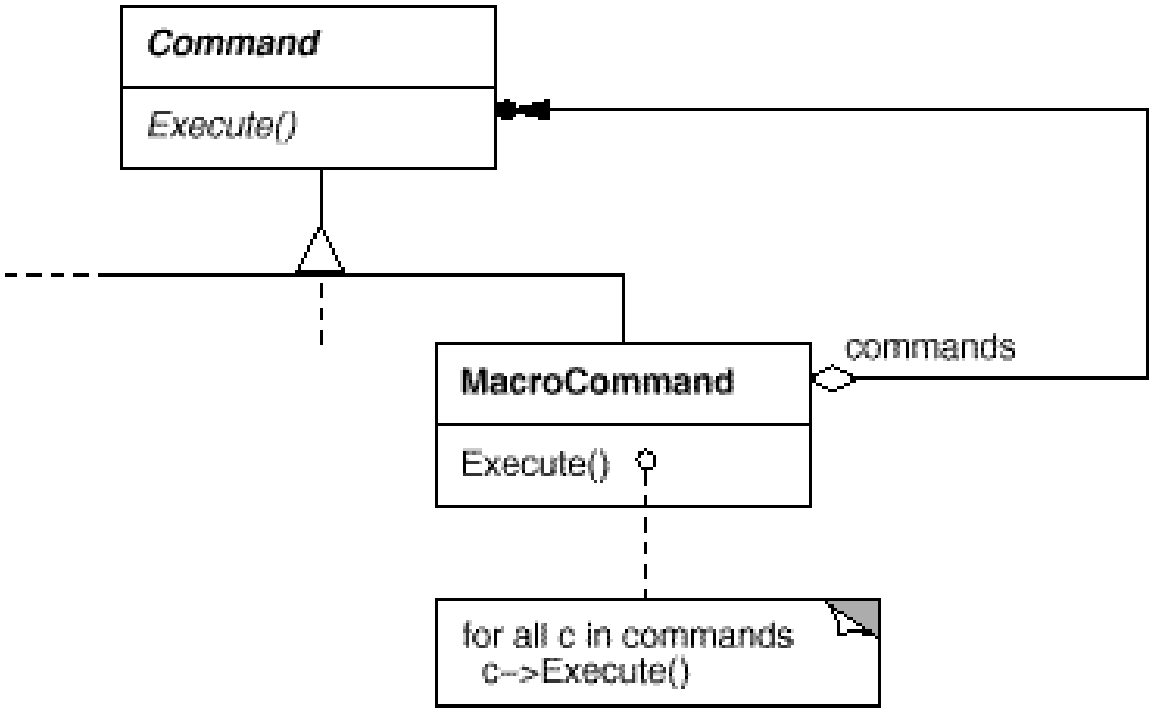
Primer Composite obrasca



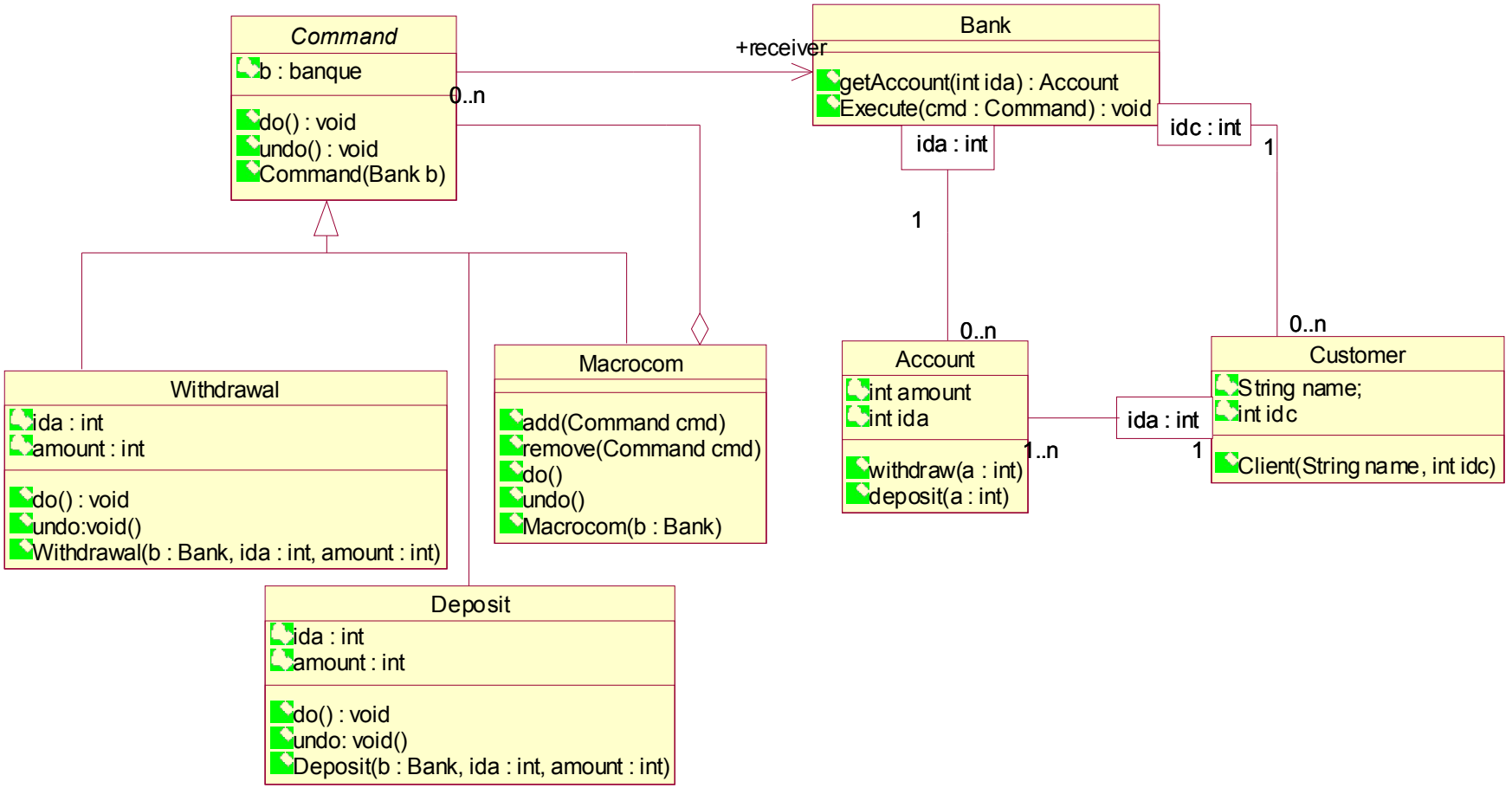
Struktura Composite obrasca

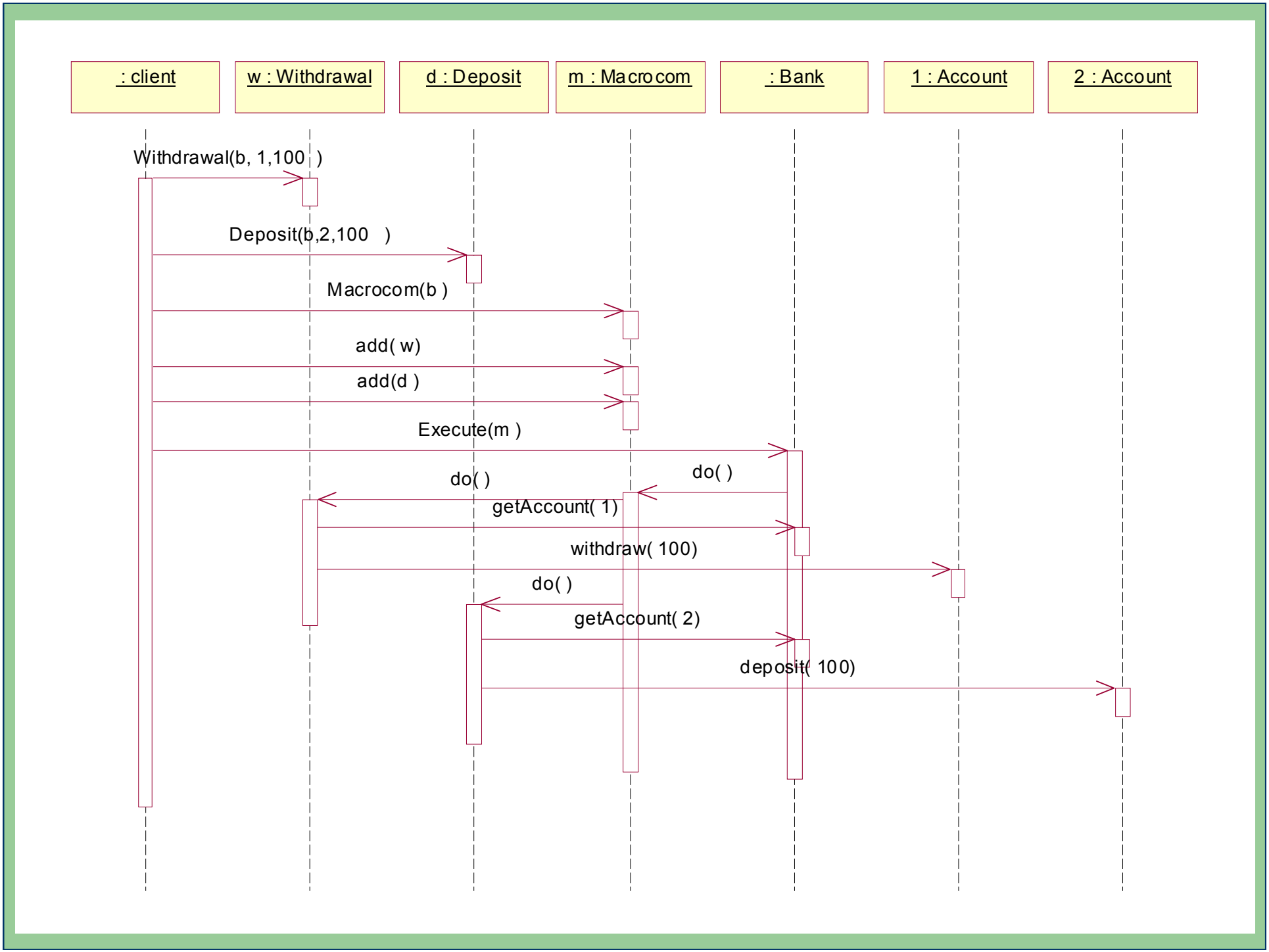


Primena Composite obrasca na Command obrazac

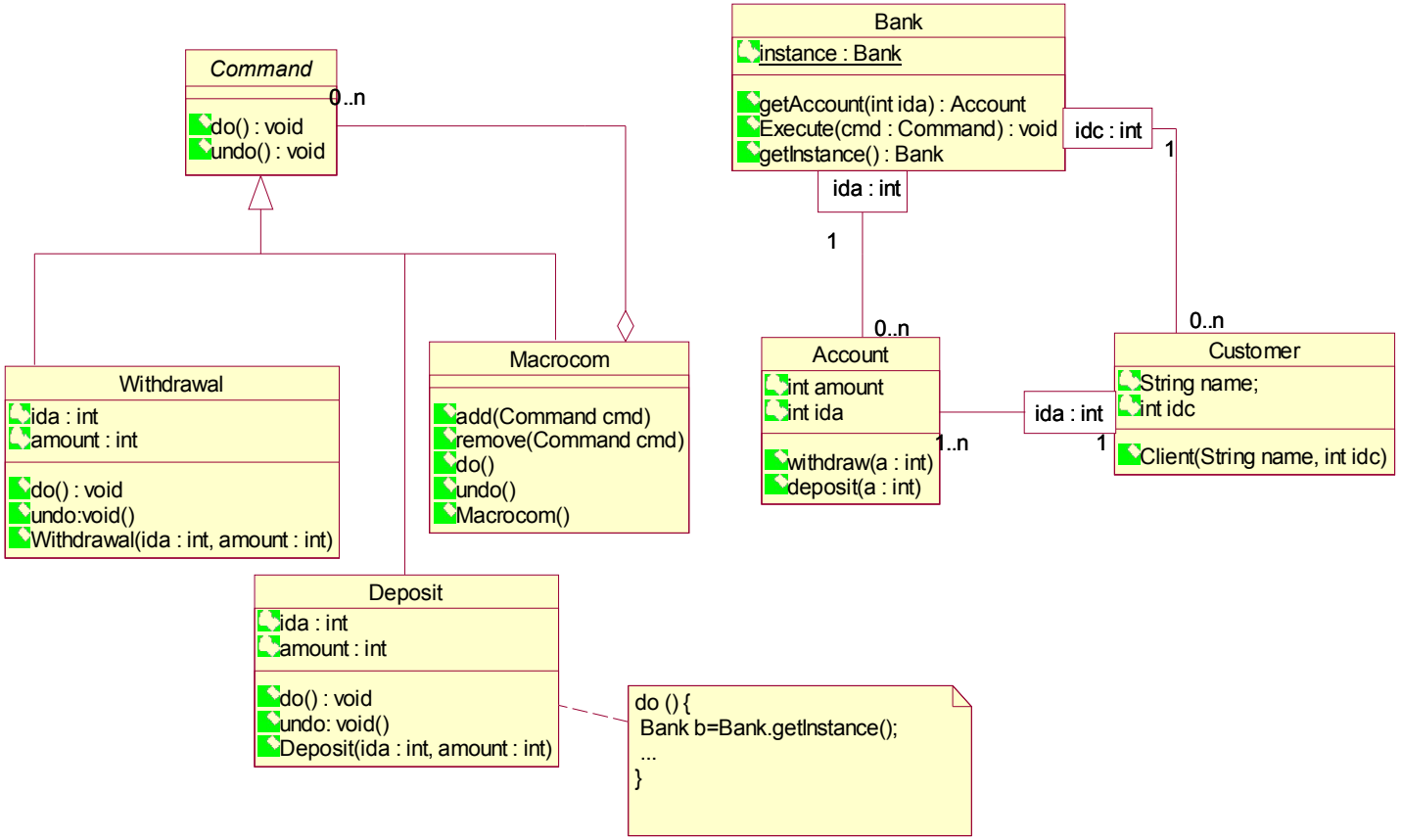


Primena Composite obrasca




: client
w : Withdrawal
d : Deposit
m : Macrocom
: Bank
1 : Account
2 : Account
Withdrawal(b, 1,100)
Deposit(b,2,100)
Macrocom(b)
add(w)
add(d)
Execute(m)
do( )
do( )
getAccount(1)
withdraw(100)
do( )
getAccount(2)
deposit(100)

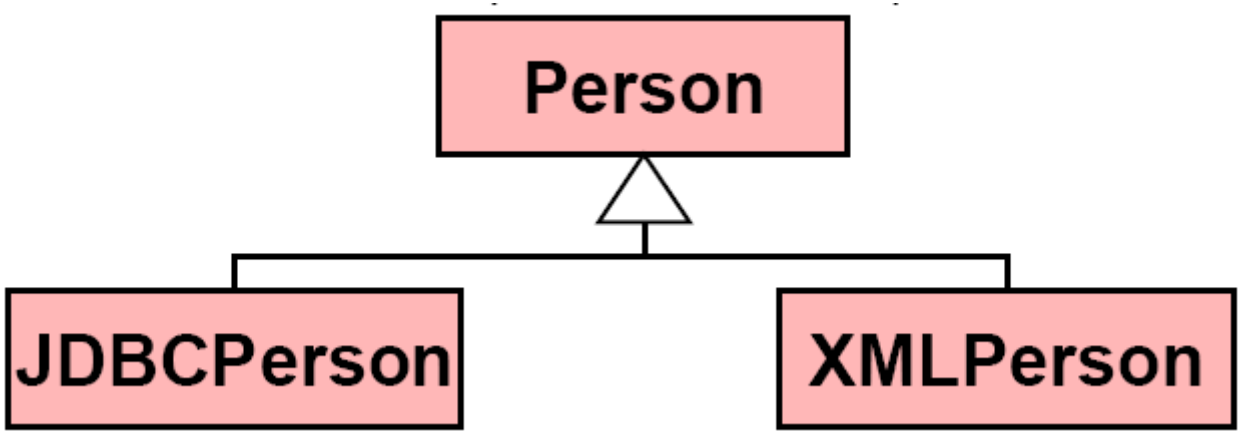
Primena Singleton obrasca



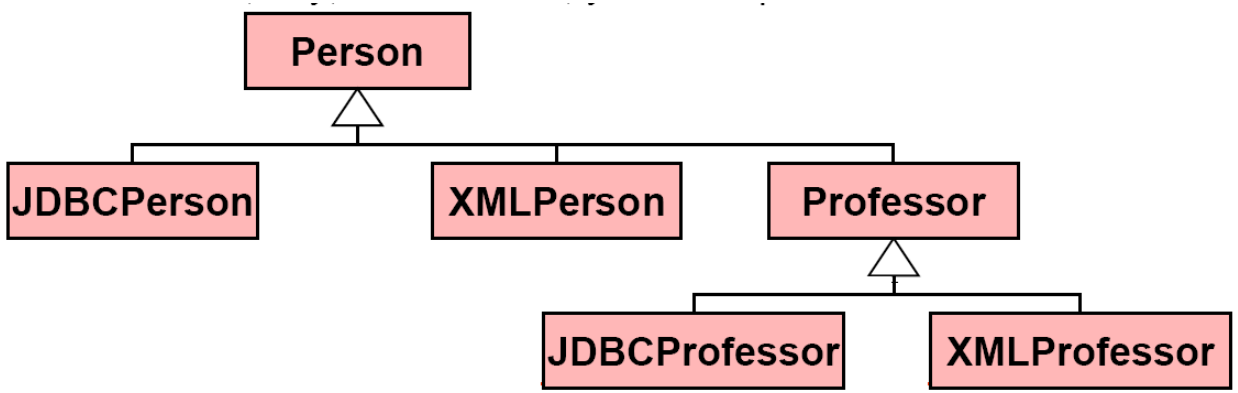
Treći primer – Hijerarhija klasa za smeštaj podataka o osobama

- Treba projektovati hijerarhiju klasa koja će omogućiti čuvanje osnovnih podataka o osobama:
  - Treba omogućiti čuvanje podataka u XML fajlu.
  - Treba omogućiti čuvanje podataka u bazi podataka.

Naivno rešenje



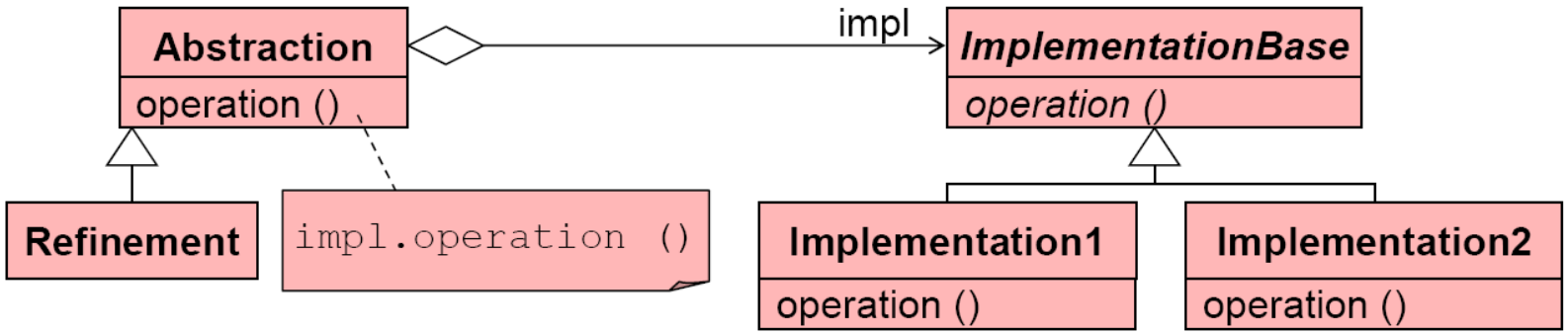
Naivno rešenje



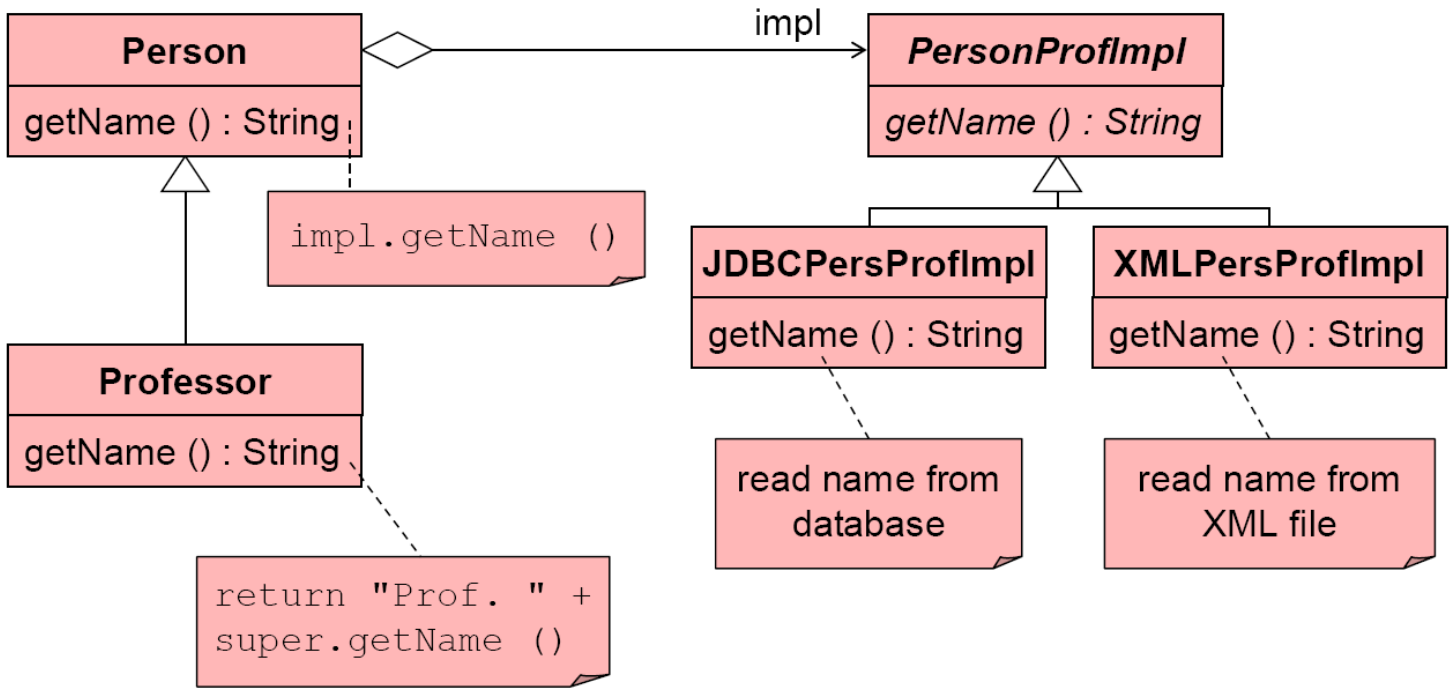
Bridge – projektni obrazac

- Povezuje razdvojene hijerarhije konceptualnih (apstraktnih) klasa i njihovih konkretnih implementacija.
- Promene u implementacionom delu ne zahtevaju promenu koda na klijentskoj strani.
- Implementacija je kompletno sakrivena od klijenta.

Bridge obrazac



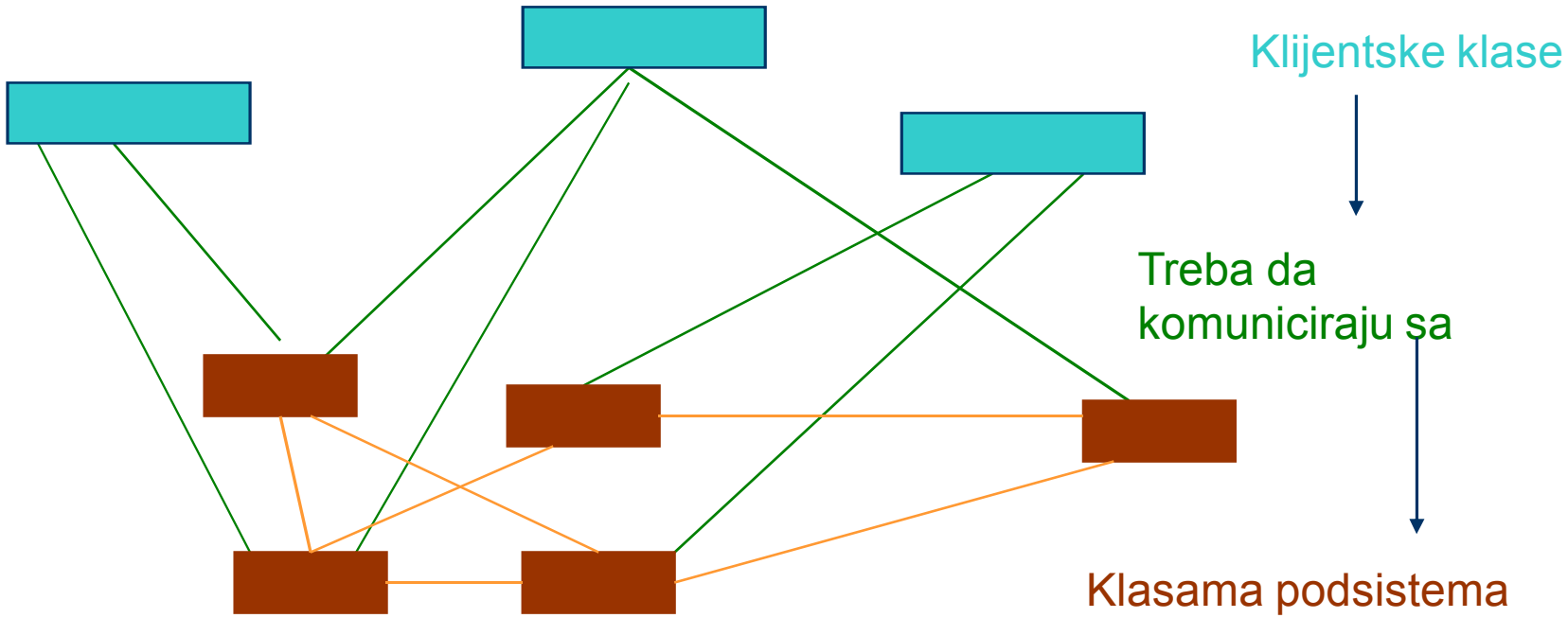
Rešenje primenom Bridge obrasca



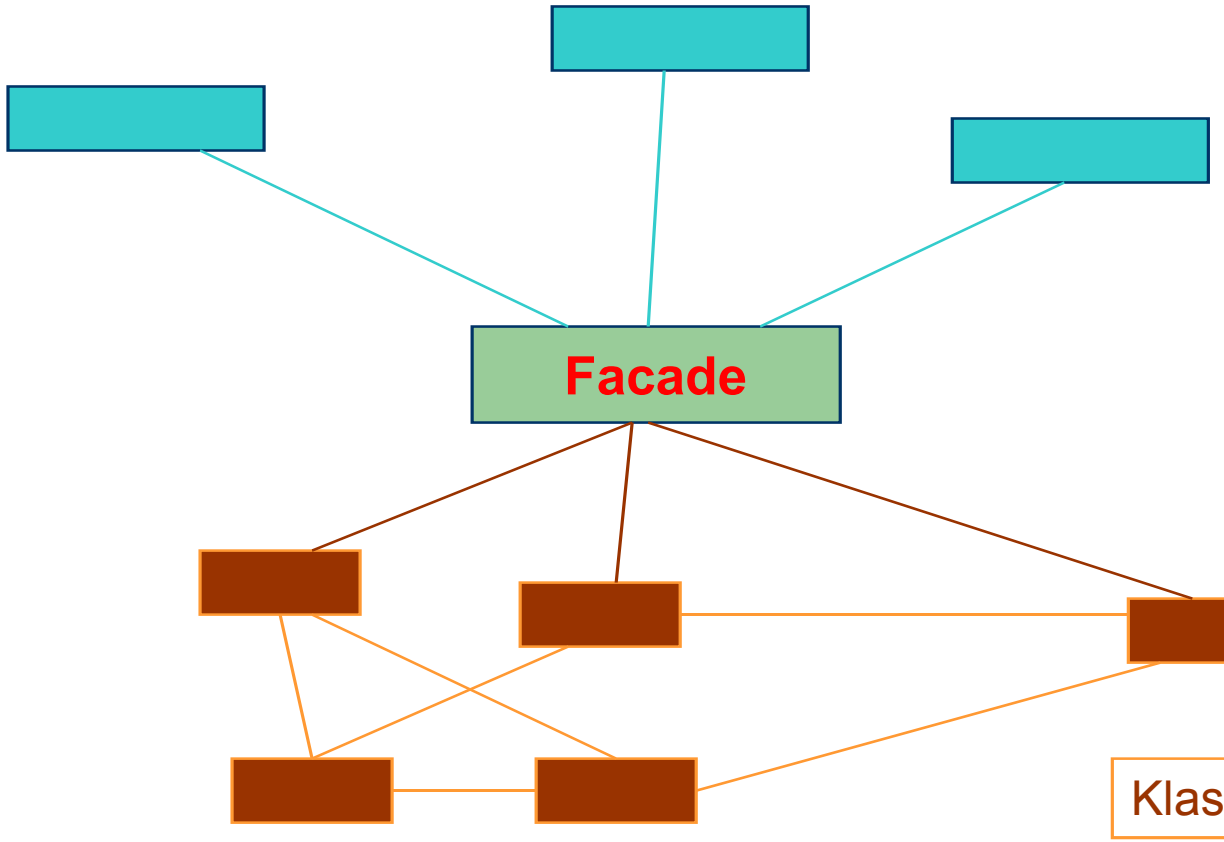
Četvrti primer – Hijerarhija klasa za realizaciju kompajlera

- Treba projektovati hijerarhiju klasa za realizaciju kompajlera.

Facade – projektni obrazac - problem



Facade – projektni obrazac



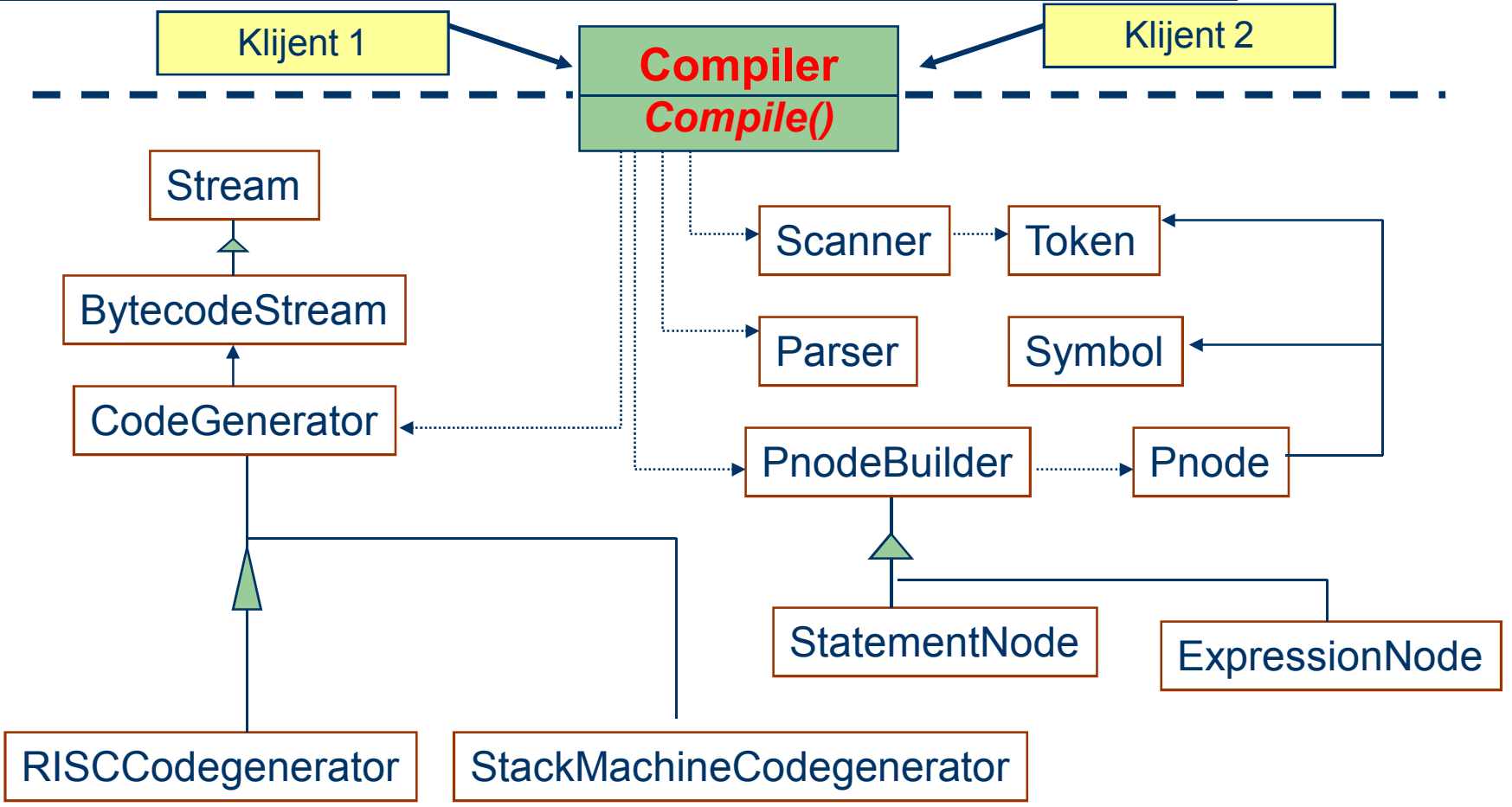
Klijentske klase

Klase podsistema

Facade – projektni obrazac - rešenje

- Razdvaja podsistem od klijenata koji koriste usluge podsistema.
- Obezbeđuje jednostavan interfejs ka klijentima.
- Implementacija je kompletno sakrivena od klijenta.
- Omogućava jednostavniju organizaciju sistema po slojevima (lejerima)

Facade obrazac primenjen u realizaciji kompajlera



Anti-obrasci (anti-patterns)

- Anti-obrasci su projektni obrasci koje bi trebalo izbegavati kod razvoja softvera.

Primeri anti-obrazaca

- Kodiranje pomoću izuzetaka
- DLL pakao
- Hard-kodiranje
- Špageti kod (nestruktuiranost)
- Zavisnost od specijalnih biblioteka
- ...