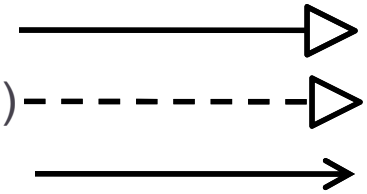
Dizajn paterni

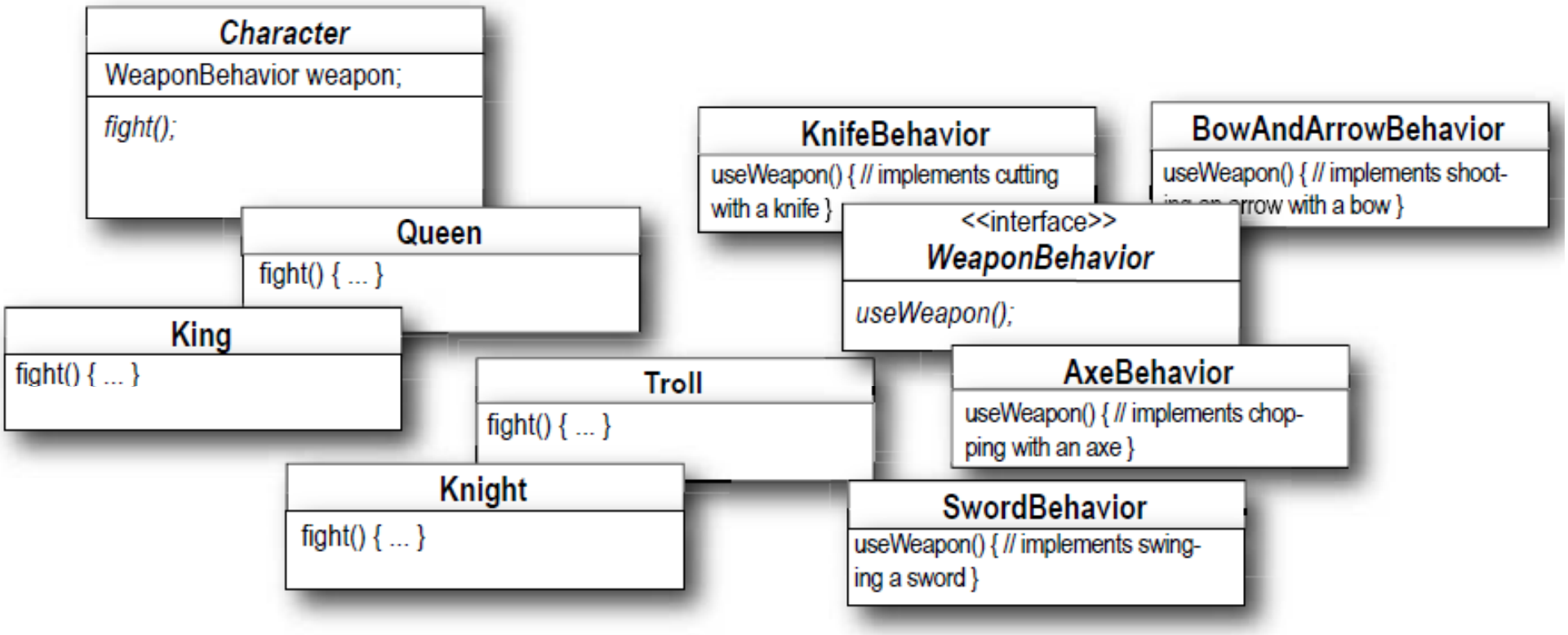
Adapter, Singleton, Facade Method, Observer i Composite

Design Puzzle – problem za zagrevanje

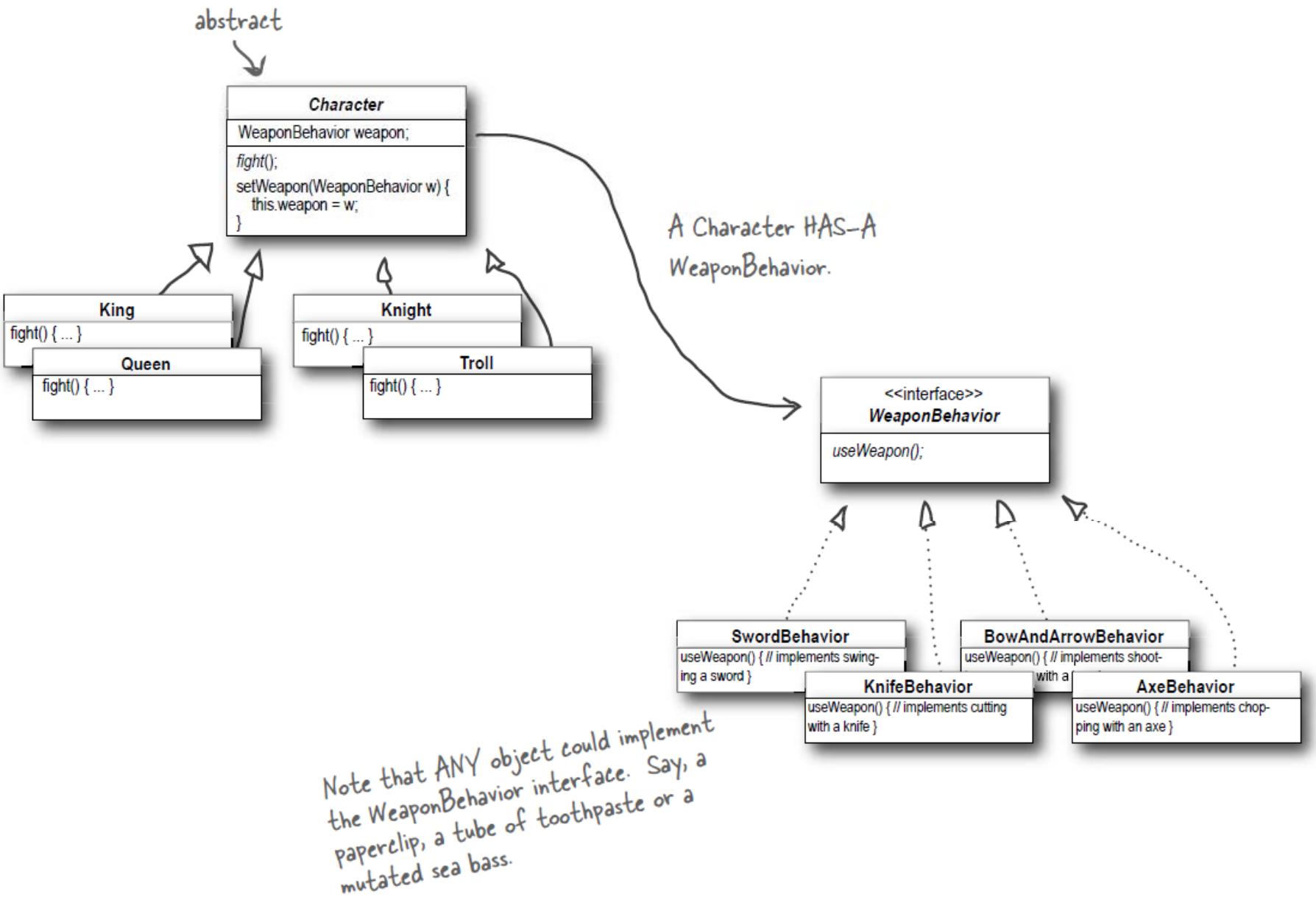
- Dat je skup klasa i interfejsa koji predstavljaju akcionu avanturističku igru. Klase koje se u njemu nalaze predstavljaju likove iz igre kao i ponašanje oružja koje likovi mogu koristiti u igri. Svaki karakter može koristiti samo jedno oružje u jednom trenutku, ali može promeniti oružje u toku igre. Vaš zadatak je da:
  - Složite klase,
  - Identifikujete 1 abstraktnu klasu, 1 interfejs i 8 klasa,
  - Nacrtate relacije između klasa, i to:
    - Nasleđivanje (extends)
    - Implementacija (impements)
    - Ima (has-a)



  - Smestiti metodu setWeapon() u odgovarajuću klasu




```
setWeapon(WeaponBehavior w) {
    this.weapon = w;
}
```


abstract
Character
WeaponBehavior weapon;
fight();
setWeapon(WeaponBehavior w) {
this.weapon = w;
}
A Character HAS-A WeaponBehavior.
King
fight() { ... }
Queen
fight() { ... }
Knight
fight() { ... }
Troll
fight() { ... }
<<interface>>
WeaponBehavior
useWeapon();
SwordBehavior
useWeapon() { // implements swinging a sword }
BowAndArrowBehavior
useWeapon() { // implements shooting a
KnifeBehavior
useWeapon() { // implements cutting with a knife }
AxeBehavior
useWeapon() { // implements chopping with an axe }
Note that ANY object could implement the WeaponBehavior interface. Say, a paperclip, a tube of toothpaste or a mutated sea bass.

Šta je dizajn patern?

- Dizajn patern
  - Rešenje učestalog problema u dizajnu,
  - Rešenje je dokumentovano u abstraktnoj i kompaktnoj formi i sadrži:
    - Opis problema,
    - Kontekst u kome se javlja,
    - Opis dobrog rešenja

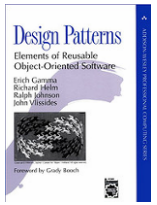
Šta je dizajn patern?

- Dizajn patern ima najmanje i obavezno 4 osnovna dela :
  - Ime
  - Problem
  - Rešenje
  - Posledice
- Jezička i implementaciona nezavisnost
- Mikro arhitektura
- Usklađenost sa postojećim metodologijama
- Nema mehaničke primene
  - Programer mora prevesti rešenje u konkretne pojmove u kontekstu aplikacije

Zbog čega su nam potrebni dizajn paterni?

- ✧ Višestruka upotreba znanja
  - Problemi nisu uvek jedinstveni. Višestruko korišćenje prethodnog iskustva može biti korisno.
  - Paterni nam daju savete “gde da tražimo probleme”.
- ✧ Uspostavljaju zajedničku terminologiju
  - Lakše je reći: “Ovde nam treba Facade “.
- ✧ Obezbeđuju veći stepen apstrakcije
  - Oslobađa nas potrebe za rad sa previše detalja u ranim fazama

Istorija dizajn paterna

Christopher Alexander *The Timeless Way of Building* *A Pattern Language: Towns, Buildings, Construction*	Architecture	1970’
Gang of Four (GoF) *Design Patterns: Elements of Reusable Object-Oriented Software* 	Object Oriented Software Design	1995’
Mnogi autori	Druge oblasti: HCI, Organizational Behavior, Education, Concurent Programming…	2007’

Struktura dizajn paterna*

- Ime paterna i klasifikacija
  - Sažeto izlaže suštinu.
- Namena
  - Kratak iskaz o tome šta patern radi.
- Motivacija
  - Scenario koji ilustruje problem projektovanja i način na koji strukture klasa i objekata u paternu rešavaju taj problem.
- Primenjivost
  - Na koje situacije se patern može primeniti.

*prema GoF

Struktura dizajn paterna*

- Struktura
  - Grafiki prikaz klasa u paternu.
- Učesnici
  - Klase i/ili objekti koji učestvuju u paternu kao i njihove odgovornosti.
- Saradnja
  - Kako učesnici sarađuju da bi izvršili svoje odgovornosti.
- Posledice
  - Koje su prednosti i nedostaci korišćenja paterna
- Implementacija
  - Saveti i tehnike za implementaciju paterna

Tri tipa GoF paterna prema nameni

- ✧ Creational (gradivni) paterni:
  - Bave se postupkom inicijalizacije i konfigurisanja objekata.
- ✧ Structural (strukturni) paterni:
  - Bave se sastavljanjem klasa i objekata.
- ✧ Behavioral (ponašanje) paterni:
  - Opisuju kako klase ili objekti međusobno utiču jedni na druge i kako dele odgovornosti.

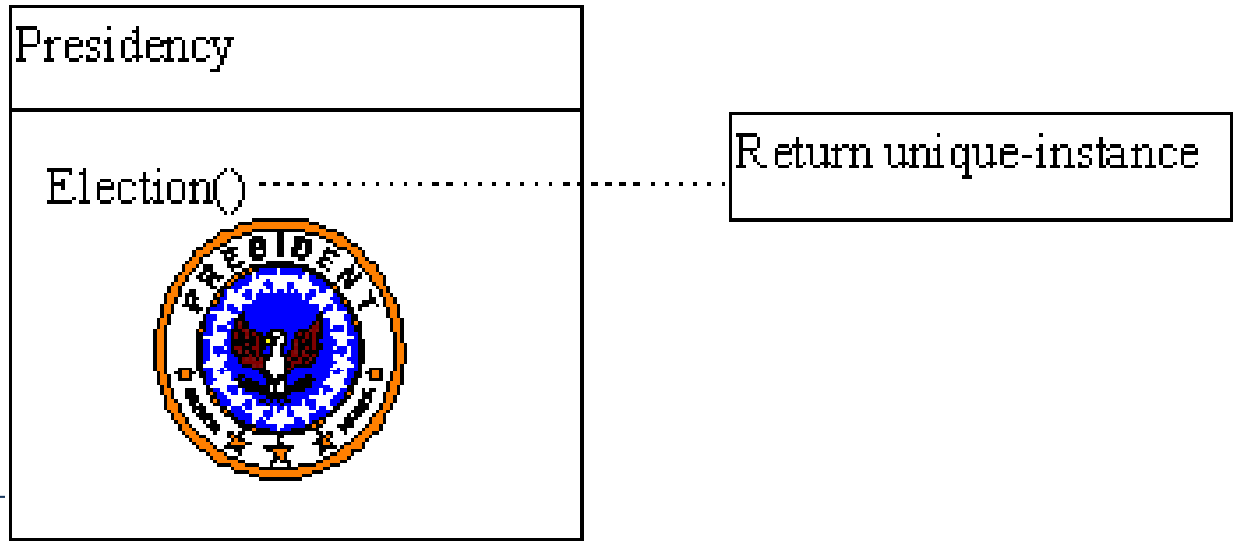
Gradivni paterni

SINGLETON


I tell ya she's ONE OF A KIND. Look at the lines, the curves, the body, the headlights!
You talkin' to me or the car? Oh, and when can I get my oven mitt back?

Singleton

- Singelton patern obezbeđuje da klasa ima samo jednu instancu i daje mu globalnu tačku pristupa.
- Primer: Kabinet predsednika je Singleton. Država definiše sredstva kojima se bira predsednik i ograničava pojam kabineta. Kao rezultat, može postojati samo jedan aktivan predsednik u jednom vremenskom trenutku. Bez obzira na lični identitet, pojam “Predsednik države” je globalna tačka pristupa koja identifikuje osobu u kabinetu.



Singleton

- Namena
- Obezbeđuje da klasa ima samo jedan primerak i daje mu globalnu tačku pristupa.

Problem

Za neke klase je važno da imaju samo jednu instancu. Mada u sistemu može da bude mnogo štampača, trebalo bi da postoji samo jedan spuler za štampanje. Trebalo bi da postoji samo jedan sistem datoteka i jedan upravljač prozorima.

Singleton

- Rešenje
- Rešenje je da se sama klasa učini odgovornom za čuvanje zapisa o svojoj sopstvenoj instanci.
- Klasa može obezbediti da se ne može kreirati druga instanca (sprečavanjem zahteva za kreiranje novih objekata), kao i način za pristup kreiranoj instanci.

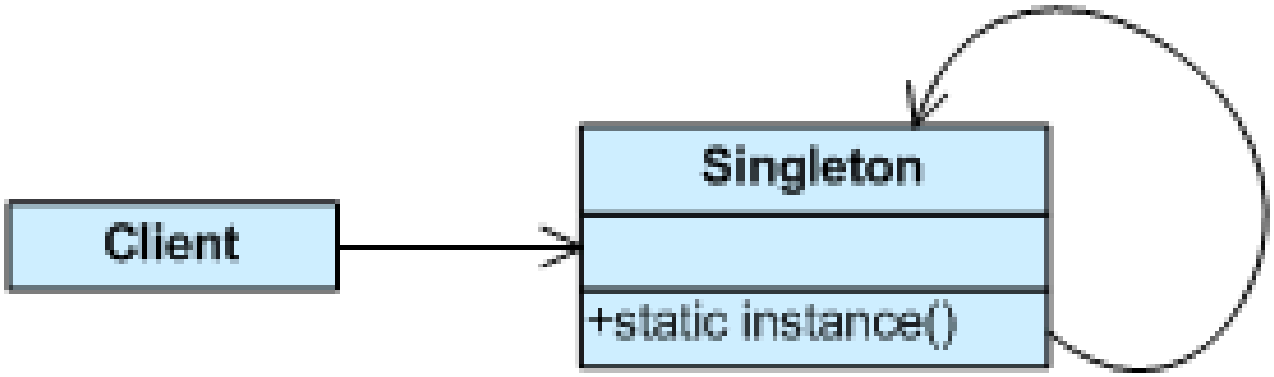
- Koraci:
- Učiniti klasu objekta sa jedinstvenom instancom odgovornom za kreiranje, inicijalizaciju i pristup.
- Deklarisati instancu kao privatnu statičku promenljivu.
- Obezbediti javnu statičku funkciju koja sadrži sav inicijalizacioni kod i obezbeđuje pristup instanci.

Singleton

- Primenljivost
- Singleton upotrebite kada:
- Treba da postoji tačno jedan primerak neke klase i on mora da bude dostupan sa dobro poznate tačke pristupa.
- Jedini primerak treba da se proširuje potklasama, a klijentima treba omogućiti da koriste prošireni primerak bez izmene postojećeg koda.

Singleton

✧ Struktura



Saradnja

- Klijenti pristupaju paternu Singleton jedino putem njegove operacije Instance().

Primer

```
class USTax {
        private static USTax instance;
        private USTax():
        public static USTax getInstance () {
                if (instance== null)
                        instance= new USTax();
                return instance;
                }
}
```

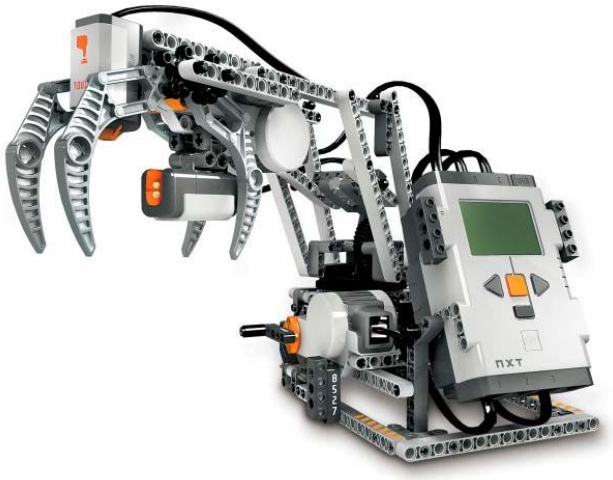
Singleton

- Posledice
- Kontrolisani pristup jedinom objektu
- Smanjen prostor imena (nema globalnih promenljivih u kojima se čuvaju jedini primerci)
- Omogućava unapređivanje operacija i predstavljanja
- Dozvoljava promenljivi broj primeraka
- Fleksibilniji od operacija klase

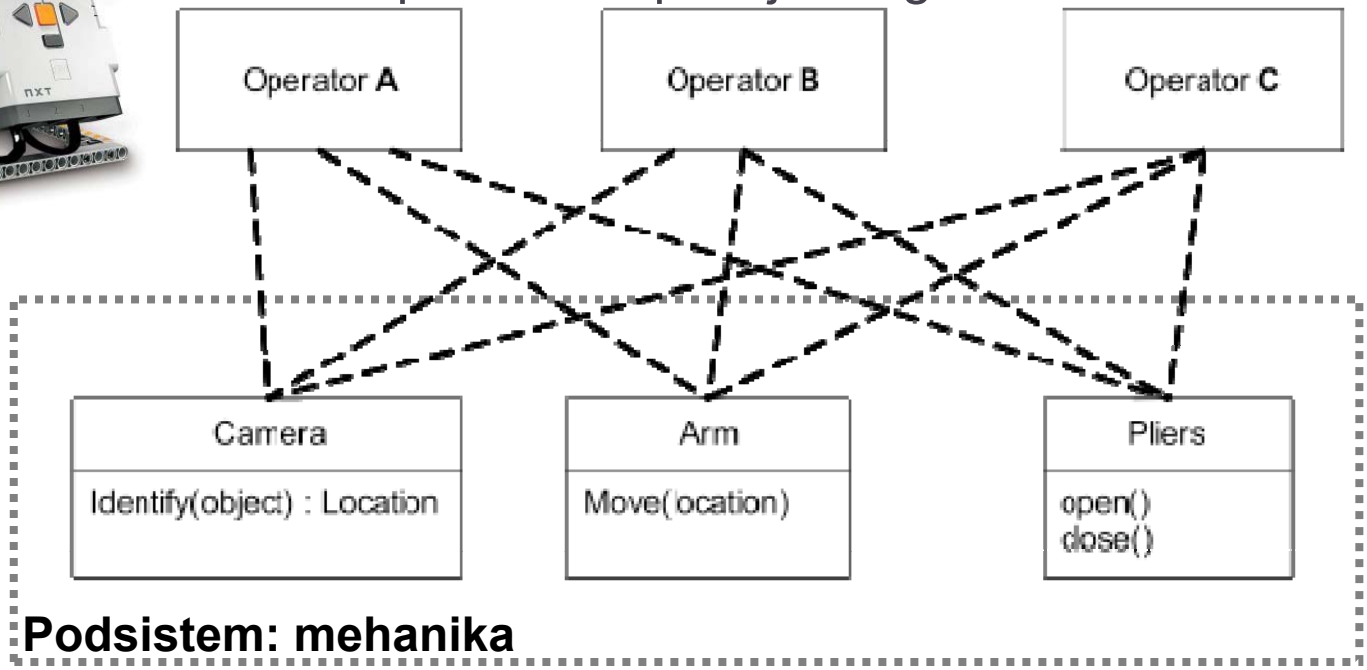
Strukturni paterni

FACADE

Facade



- Robot ima četiri klase:
  - Camera (koja identifikuje objekte)
  - Arm (ruka koja može pomerati)
  - Pillars (koji može hvatati)
  - Operator – upravlja drugim klasama



Facade

1. Koliko je potrebno operatoru u pogledu mehanike?
2. Na primer, ukoliko želimo da identifikujemo objekat i pomerimo ga na određenu lokaciju, kod bi izgledao ovako:

```
oldLocation = Camera.identify(object);
Arm.move(oldLocation);
Pliers.close();
Arm.move(newLocation);
Pliers.open();
```

- ✧ Problem: Nema enkapsulacije
  - Operator mora da zna puno: strukturu + ponašanje
  - Preduslovi
  - Spretnost

Facade

Facade

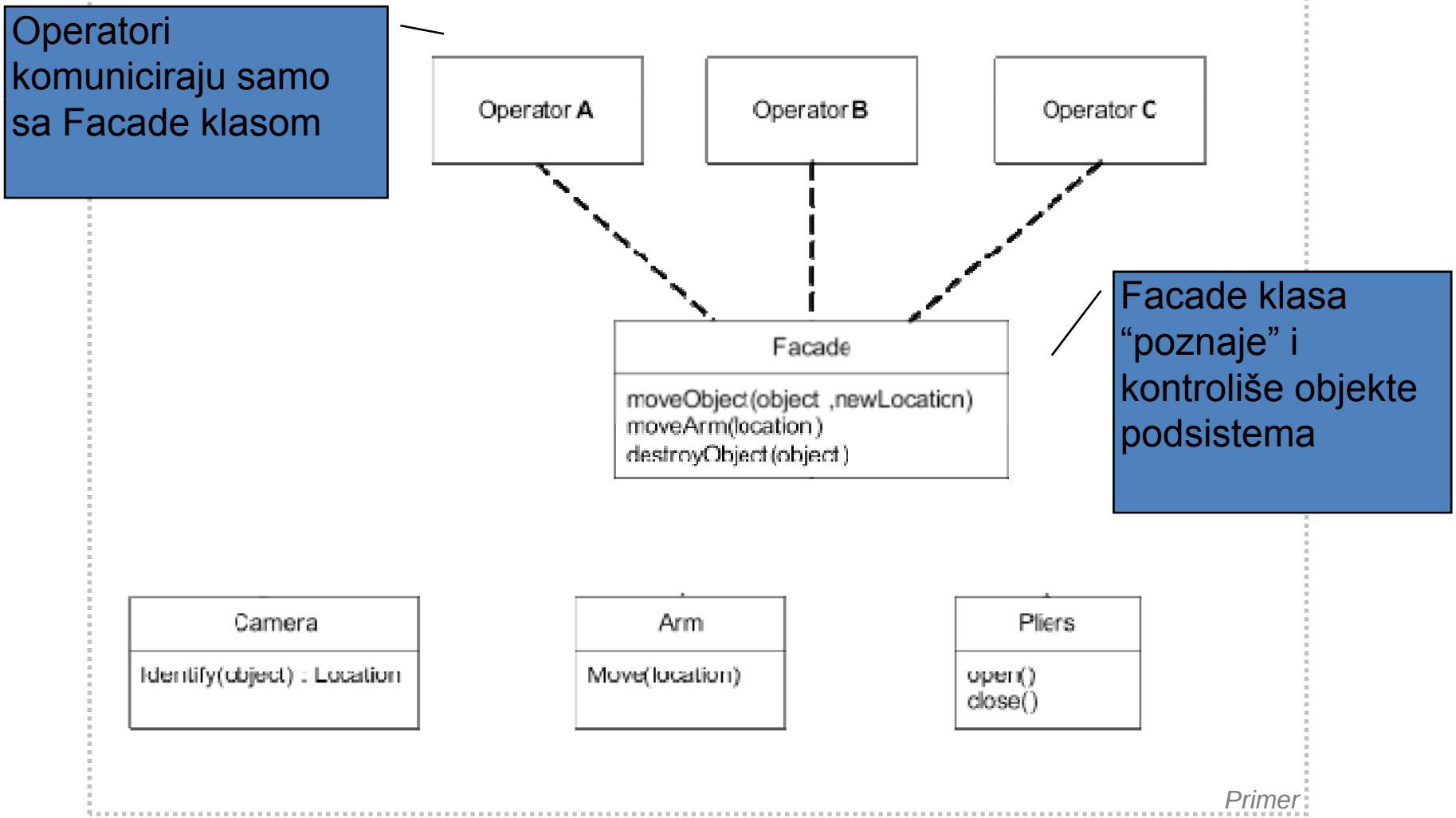
✧ Namena

- Obezbeđuje jedinstveni interfejs za skup interfejsa jednog podsistema.
- Facade definiše interfejs višeg nivoa da bi se podsistem lakše koristio.

✧ Primenjivost

- Koristite Facade u sledećim slučajevima
  - Kada složenom podsistemu hoćete da date jednostavan interfejs.
  - Kada hoćete da raslojite podsisteme. Upotrebite Facade za definisanje ulazne tačke za svaki nivo podsistema.

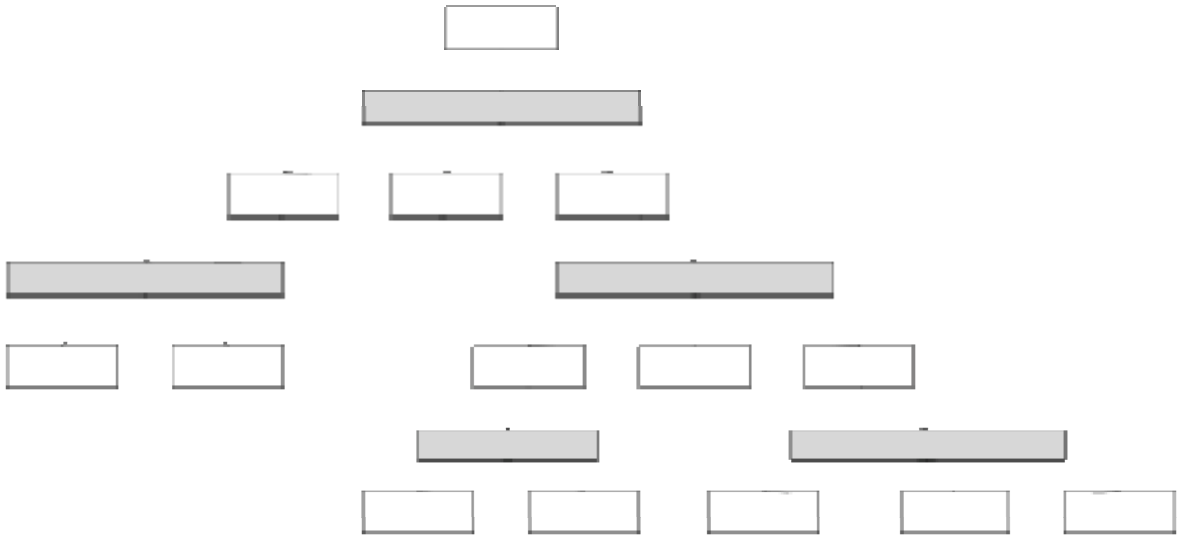
Facade - primer



Facade – sistemski nivo

✧ Enkapsulacija na višim nivoima:

- Opšti paterni za konstruisanje podsistema.
- Svaki podsistem je predstavljen facade interfejsom.
- Unutrašnji detalji su enkapsulirani.



Facade - Posledice

✧ Dobre:

- Zaklanja komponente podsistema od klijenata i tako smanjuje broj objekata sa kojima klijenti imaju posla, što olakšava upotrebu podsistema.
- Promoviše slabo vezivanje između podsistema i njegovih klijenata.
- Podstiče i olakšava slojevitu arhitekturu.

✧ Loše:

- Duplira metode.
- Slabija vidljivost funkcionalnosti podsistema.

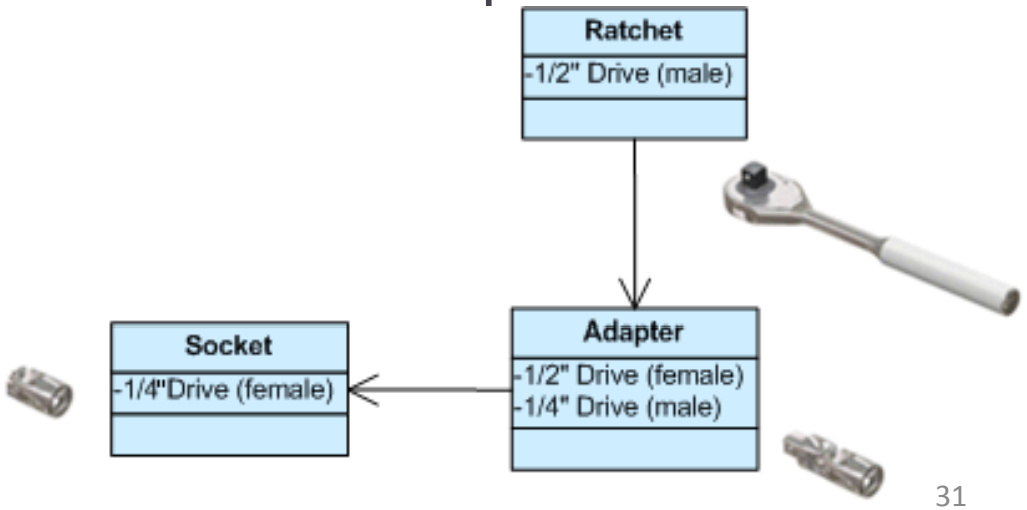
Strukturni paterni

ADAPTER


Do you think the readers are really getting the impression we're watching a horse race rather than sitting in a photo studio?
That's the beauty of our profession, we can make things look like something they're not!
You mean it's not supposed to be a football match?
Wrapped in this coat, I'm a different man!

Adapter

- Konvertuje interfejs klase u drugi interfejs koji klijenti očekuju. Adapter omogućava saradnju klasa koje inače ne bi mogle da sarađuju zbog nekompatibilnih interfejsa.
- Nasadni ključ (gedora) je primer Adaptera. Nastavci se postavljaju na nasadni ključ. Američki standardi su 1/2" i 1/4". Očigledno da 1/2“ nastavci ne mogu biti montirani na 1/4“ ključ, pa je potrebno koristiti adapter.



Adapter

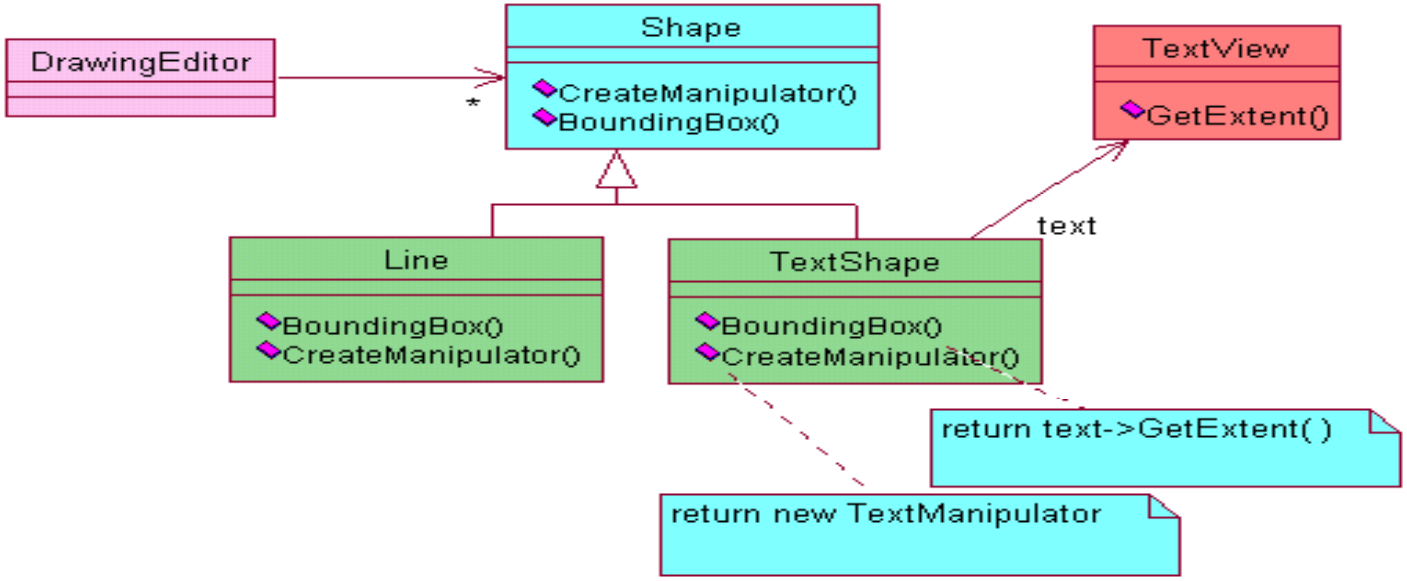
- ✧ Namena
  - Konvertuje interfejs klase u drugi interfejs koji klijenti očekuju. Adapter omogućava saradnju klasa koje inače ne bi mogle da sarađuju zbog nekompatibilnih interfejsa.
- ✧ Takođe poznat kao:
  - Wrapper (uvijač)

Adapter - problem

- Editor za crtanje (DrawingEditor) omogućava korisnicima da crtaju i raspoređuju grafičke elemente. TextShape se znatno teže implementira, pa je korisno iskoristiti postojeći tekst editor (TextView) koji nije projektovan u skladu sa klasama Shape.

Adapter - rešenje

- Jedan način rešavanja problema je da se napravi klijentska klasa (TextShape) koja će adaptirati editor (TextView) i modifikovati interfejs da odgovara aplikaciji. Ovo se može uraditi na dva načina:
  - Dodatnim izvođenjem TextView klase ili
  - Čuvanjem reference na TextView klasu unutar TextShape klase.
- Drugi metod ima prednost nad prvim jer se klijentska hijerarhija ne povećava u dubinu

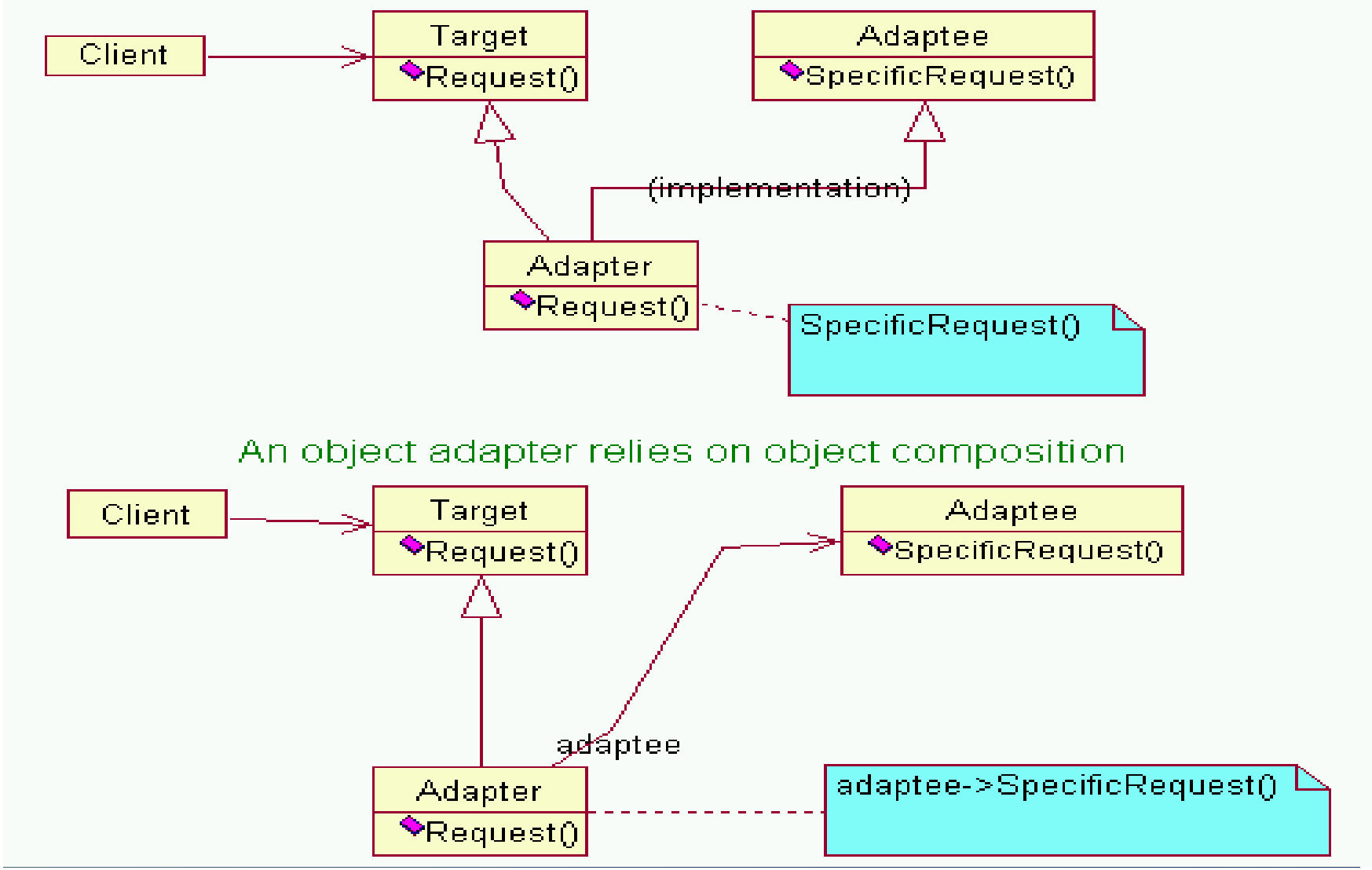


Adapter - primenjivost

✧ Koristite Adapter kada:

- Hoćete da upotrebite postojeću klasu, a njen interfejs nije usklađen sa interfejsom koji vam treba,
- Hoćete da napravite višekratno upotrebljivu klasu koja sarađuje sa nepovezanim i nepredviđenim klasama, tj. sa klasama koje mogu da imaju nekompatibilne interfejse,
- Treba da koristite nekoliko postojećih klasa, ali nije praktično da prilagođavate njihove interfejse praveći potklase svakog od njih. Objektni adapter može da prilagodi interfejs roditeljske klase.

Adapter - struktura



Adapter - posledice

- ✧ Klasni i objektni adapteri nemaju iste mane.
- ✧ Klasni adapter:
  - Prilagođava Adaptee klasu klasi Target vezivanjem za konkretnu Adaptee klasu. Zbog toga klasni adapter ne funkcioniše ako hoćemo da prilagodimo klasu i sve njene podklase.
  - Omogućava adapteru da izmeni određena ponašanja klase Adaptee, s obzirom da je Adapter podklasa klase Adaptee.
  - Uvodi samo jedan objekat, i nikakvo dodatno preusmeravanje pointera nije potrebno da bi se došlo do Adaptee klase.

Adapter - posledice

✧ Objektni adapter:

- Omogućava da jedan adapter radi sa više Adaptee objekata, tj. sa Adaptee i njegovim podklasama. Adapter može takođe dodati funkcionalnost svim Adaptee instancama odjednom.
- Teže je redefinisanje ponašanja Adaptee klase. To bi zahtevalo definisanje Adaptee podklase i povezivanje Adaptera sa njegovim podklasama a ne sa samim Adaptee klasama.

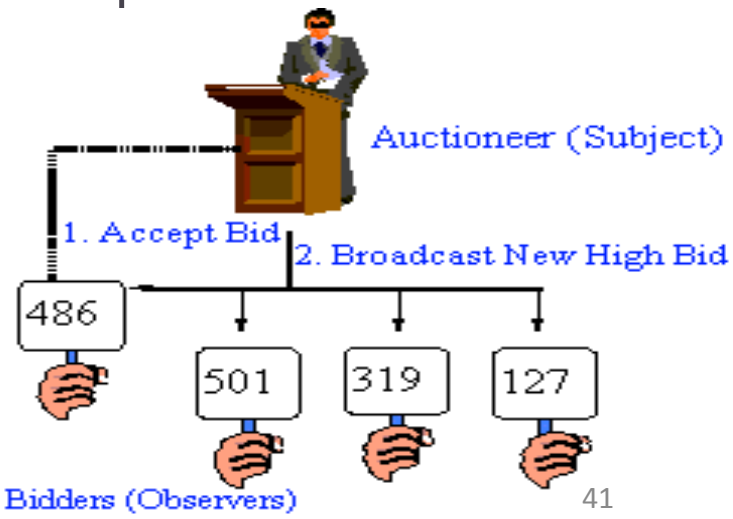
Paterni ponašanja

OBSERVER


Hey Jerry, I'm notifying everyone that the Patterns Group meeting moved to Saturday night. We're going to be talking about the Observer Pattern. That pattern is the best! It's the BEST, Jerry!

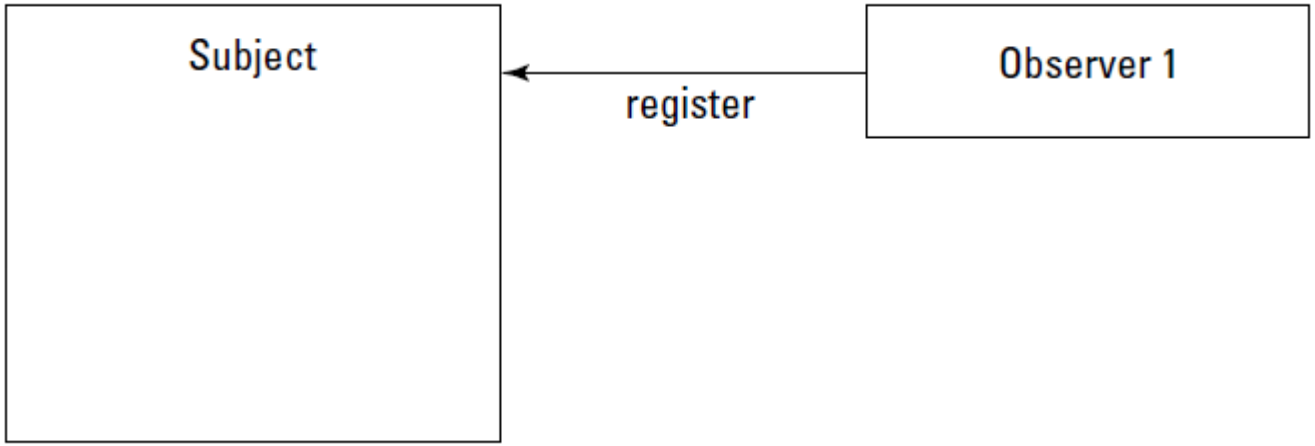
Observer

- Definiše zavisnost jedan prema više među objektima da bi prilikom promene stanja jednog objekta svi zavisni objekti bili obavešteni i automatski ažurirani.
- Neke aukcije demonstriraju ovaj patern. Svaki ponuđač poseduje tablu sa brojem koju koristi da nagovesti ponudu. Aukcionar počinje ponudu, i “posmatra” kada se tabla podigne kao prihvatanje ponude. Prihvatanje ponude menja cenu ponude, što se saopštava svim ponuđačima u formi nove ponude.



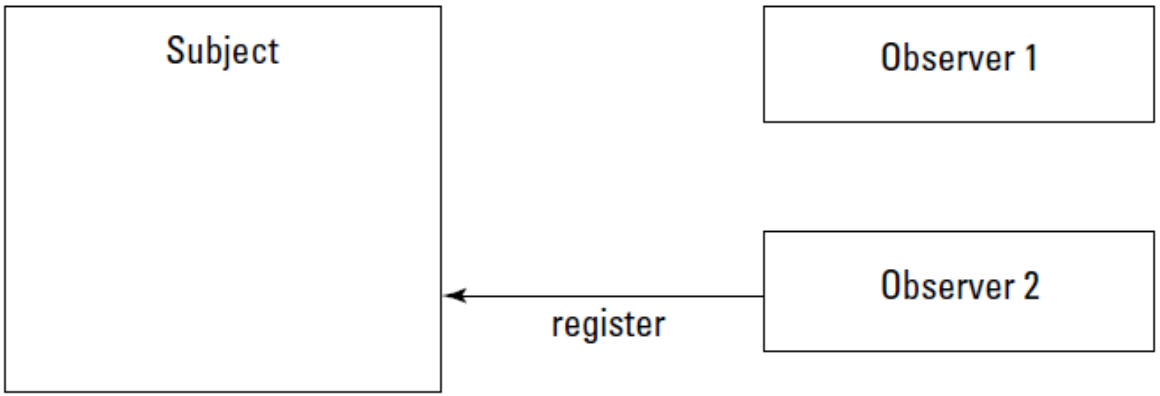
Observer

- Observer dizajn patern prosleđuje obaveštenja skupu objekata o nekom događaju koji se desio. Možete dodavati objekte posmatrače u vreme izvšenja i uklanjati ih po potrebi. Kada se desi događaj svi registrovani posmatrači će biti obavešteni.

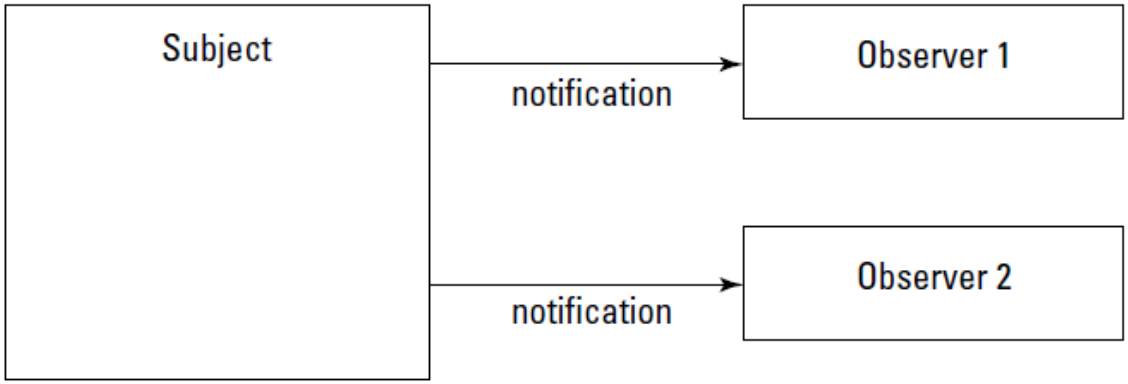


Posmatrač se može bilo kada registrovati o čemu se čuva informacija

Observer

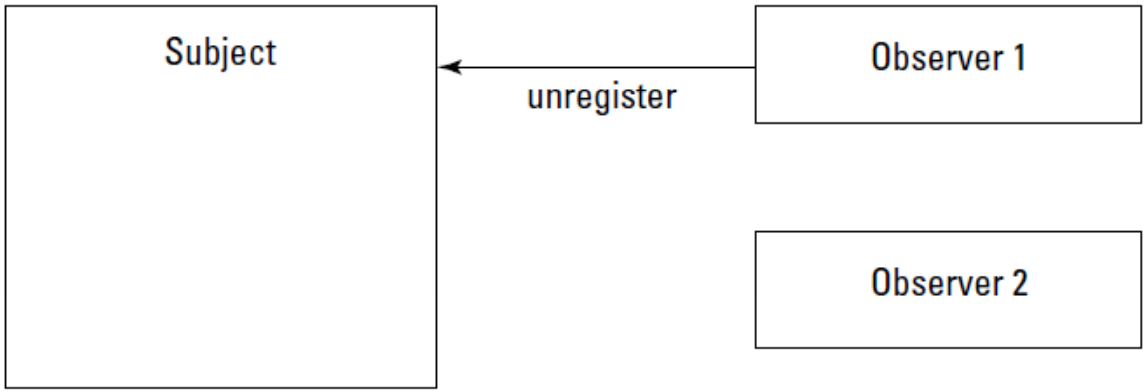


I drugi posmatrač se može registrovati

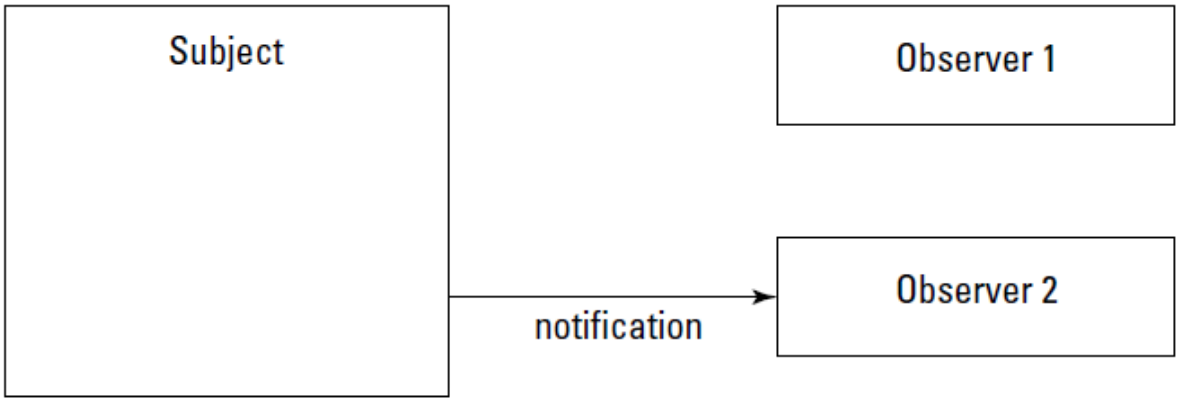


U ovom trenutku oba posmatrača se obaveštavaju o događajima

Observer



Posmatrač može bilo kada da se odjavi

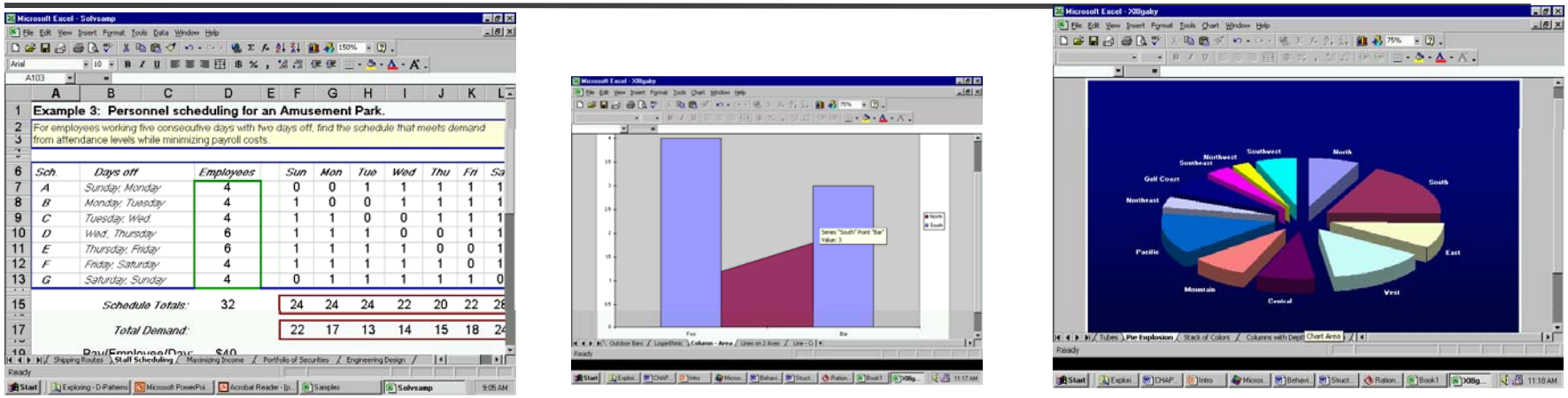


I obaveštenja mu više neće stizati.

Observer

- ✧ Namena
  - Definiše zavisnost jedan prema više među objektima da bi prilikom promene stanja jednog objekta svi zavisni objekti bili obavešteni i automatski ažurirani.
- ✧ Takođe poznat kao:
  - Zavisnici (Dependents), izdavanje-pretplata (publish-subscribe)
- ✧ Problem
  - Kako upravljati zajedničkim sporednim efektom (npr. potreba za održavanjem konzistentnosti između povezanih objekata) podele sistema u kolekciju kooperirajućih klasa bez čvrstog spajanja klasa, - što smanjuje mogućnost ponovne upotrebe klasa.

Observer - ilustracija



Promena obaveštenja

a=50%
b=30%
c=20% e.t.c

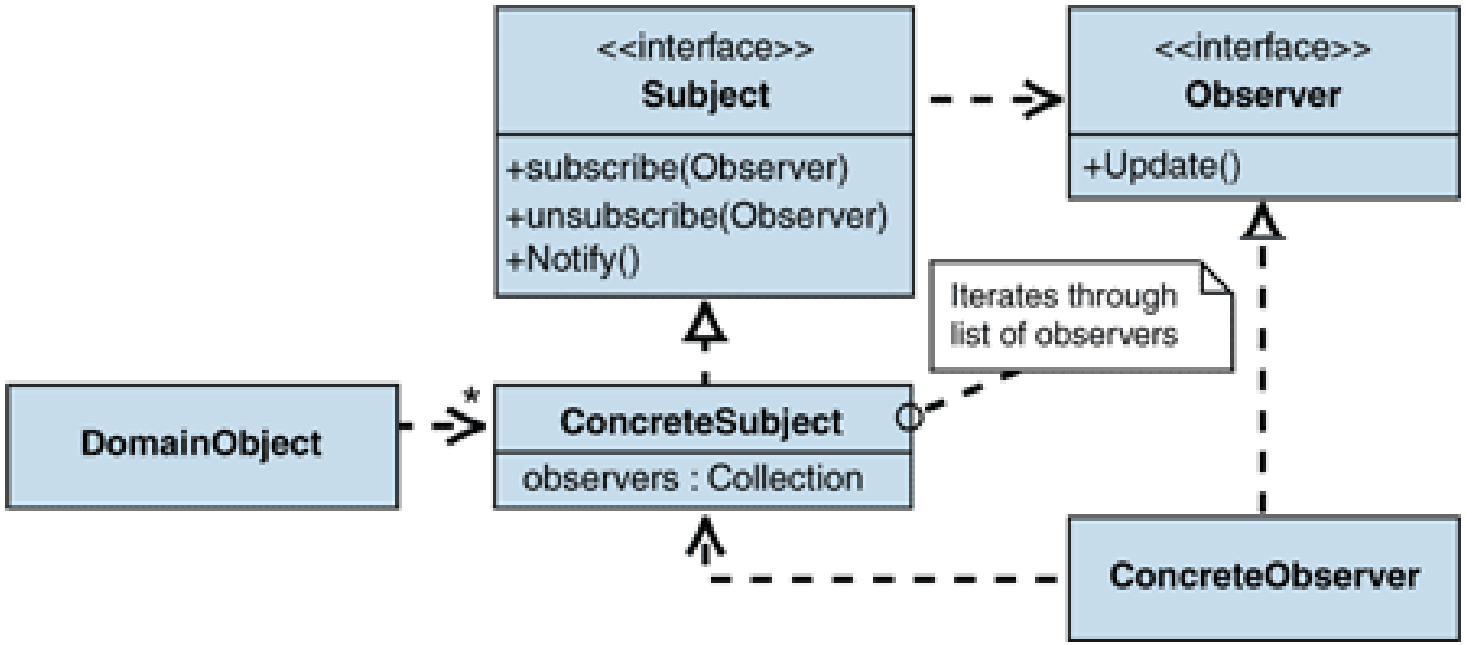
subjekat

zahtev, modifikacija

Observer - primenljivost

- ✧ Kada apstrakcija ima dva aspekta od kojih jedan zavisi od drugoga. Enkapsuliranje tih aspekata u zasebne objekte omogućava njihovo nezavisno menjanje i višekratnu upotrebu.
- ✧ Kada promena u jednom objektu zahteva promene u drugim objektima, a ne znate koliko drugih objekata treba promeniti
- ✧ Kada objektu treba omogućiti da obaveštava druge objekte bez ikakve pretpostavke o tome koji su to objekti. Drugim rečima, kada ne želite da ti objekti budu tesno vezani.

Observer - struktura



Observer - saradnja

- ConcreteSubject obaveštava posmatrače kad god dođe do promene zbog koje bi stanje posmatrača postalo nekonzistentno sa njegovim.
- Kada dobije informaciju o promeni konkretnog subjekta, objekat ConcreteObserver može subjektu da pošalje upit za informacije. ConcreteObserver koristi te informacije za usklađivanje svog stanja sa stanjem subjekta

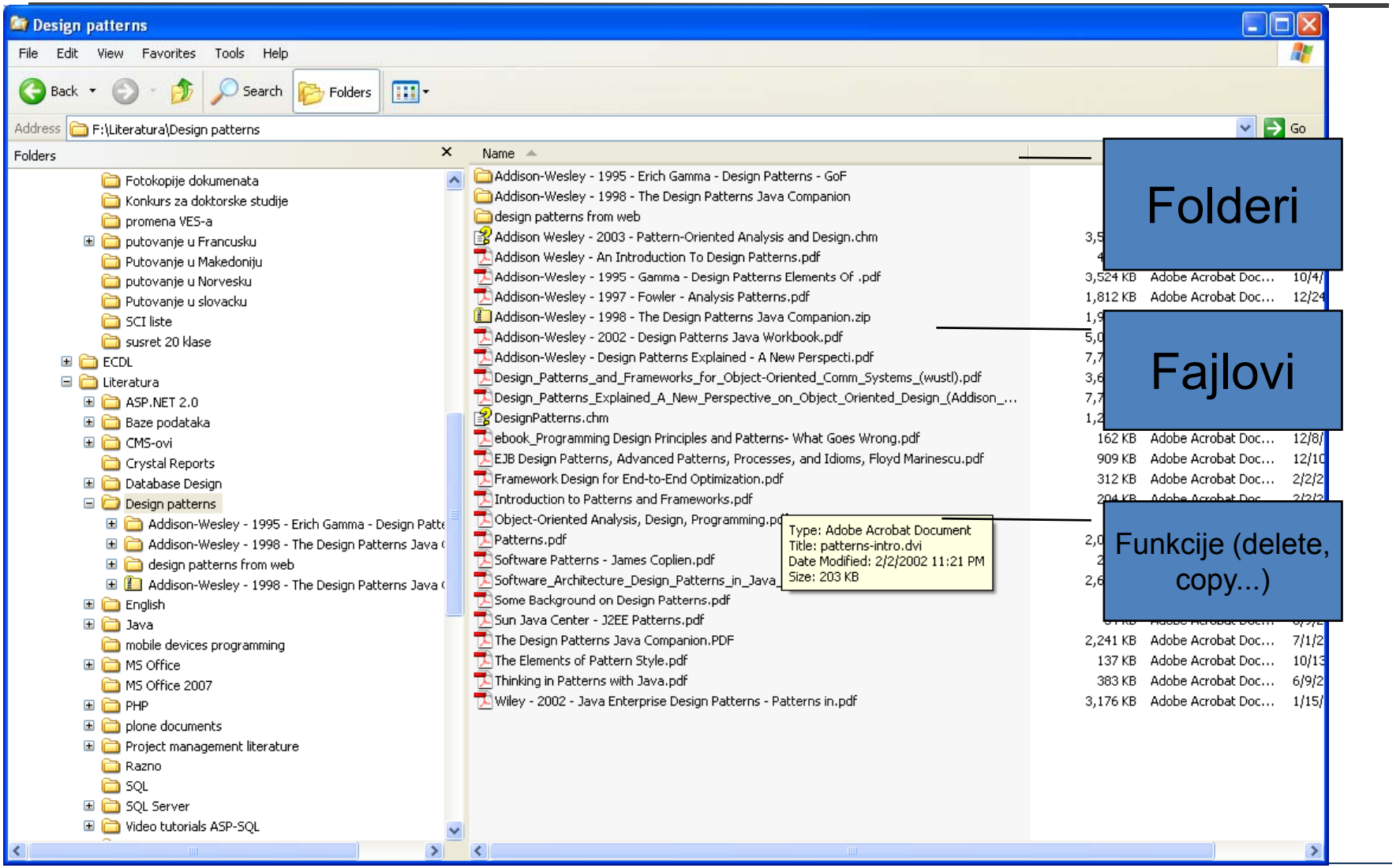
Observer - posledice

- Apstraktno vezivanje subjekta i posmatrača
- Podrška za difuznu (broadcast) komunikaciju
- Neočekivana ažuriranja (posmatrači ne znaju jedan za drugoga pa ne očekuju promenu stanja)

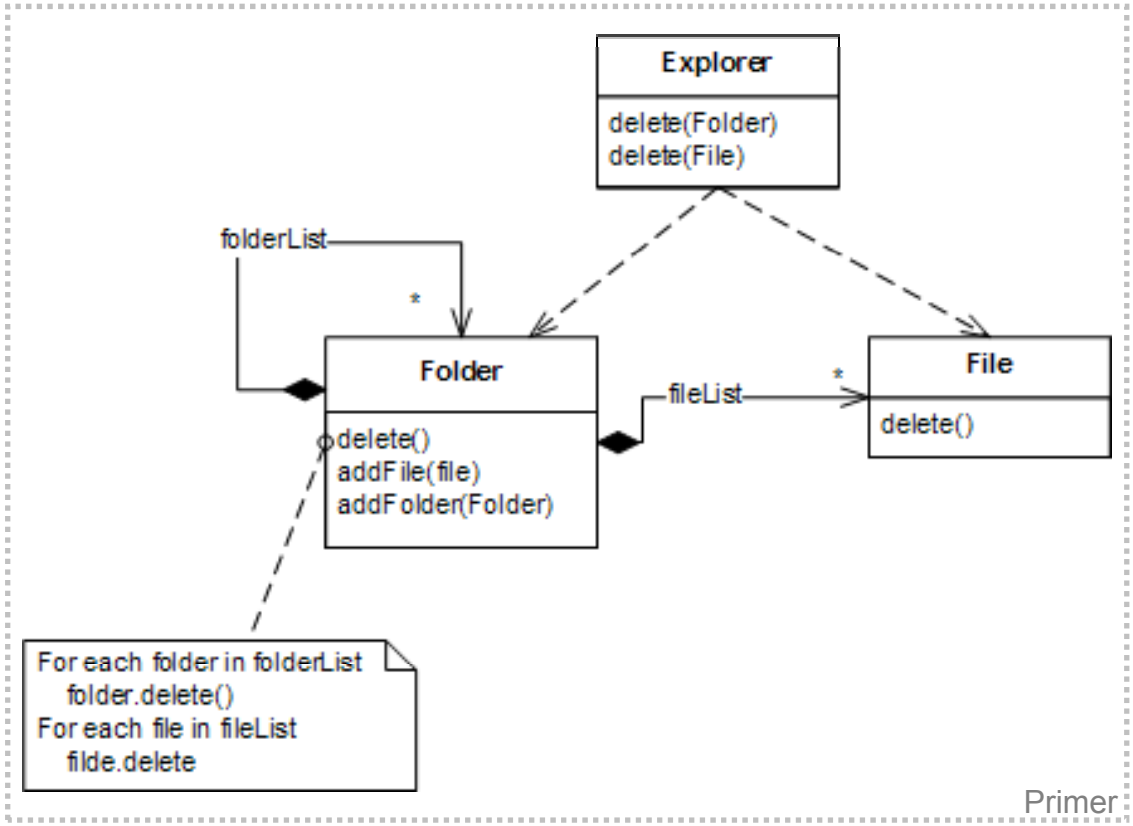
Strukturni paterni

COMPOSITE

Composite (Sastav) - problem



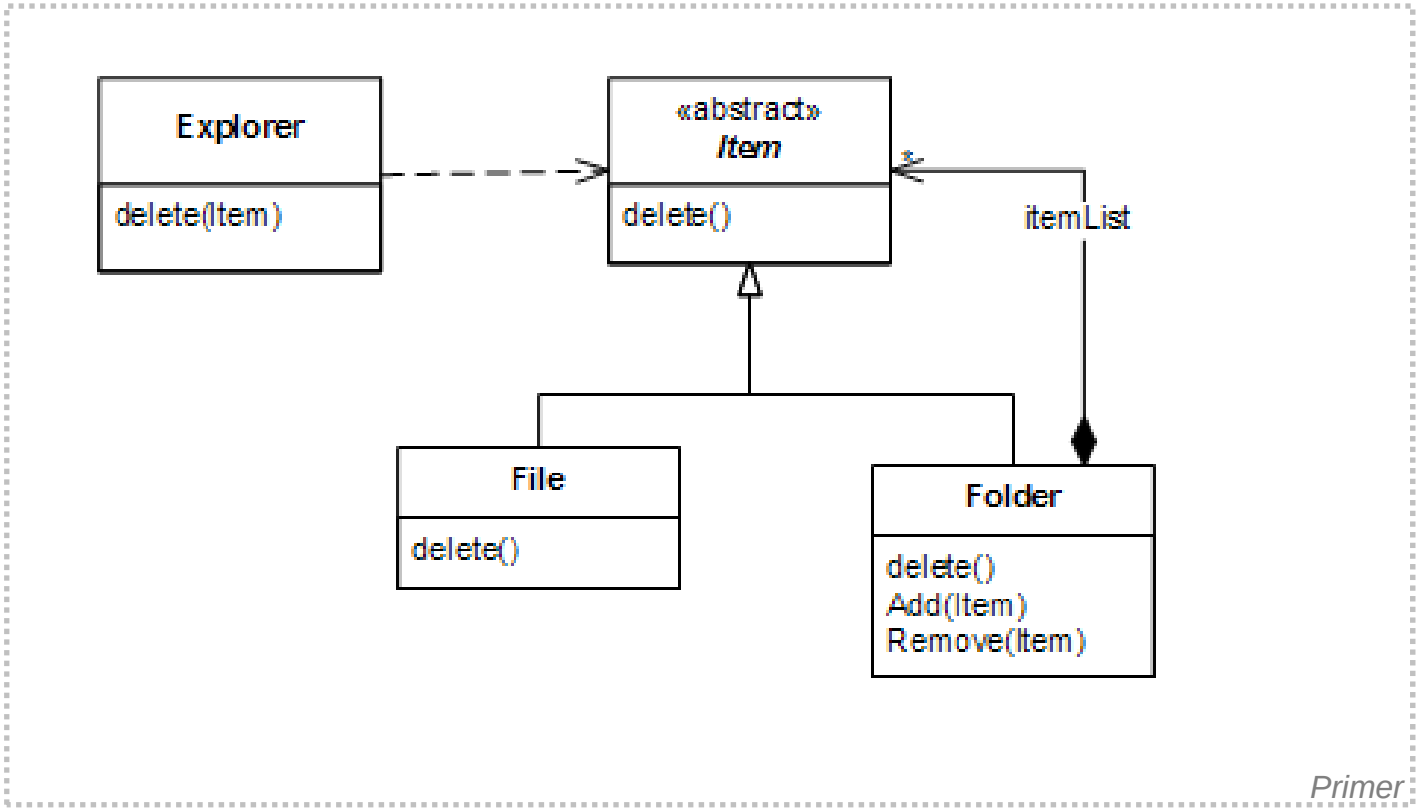
Composite (Sastav) - rešenje



Primer

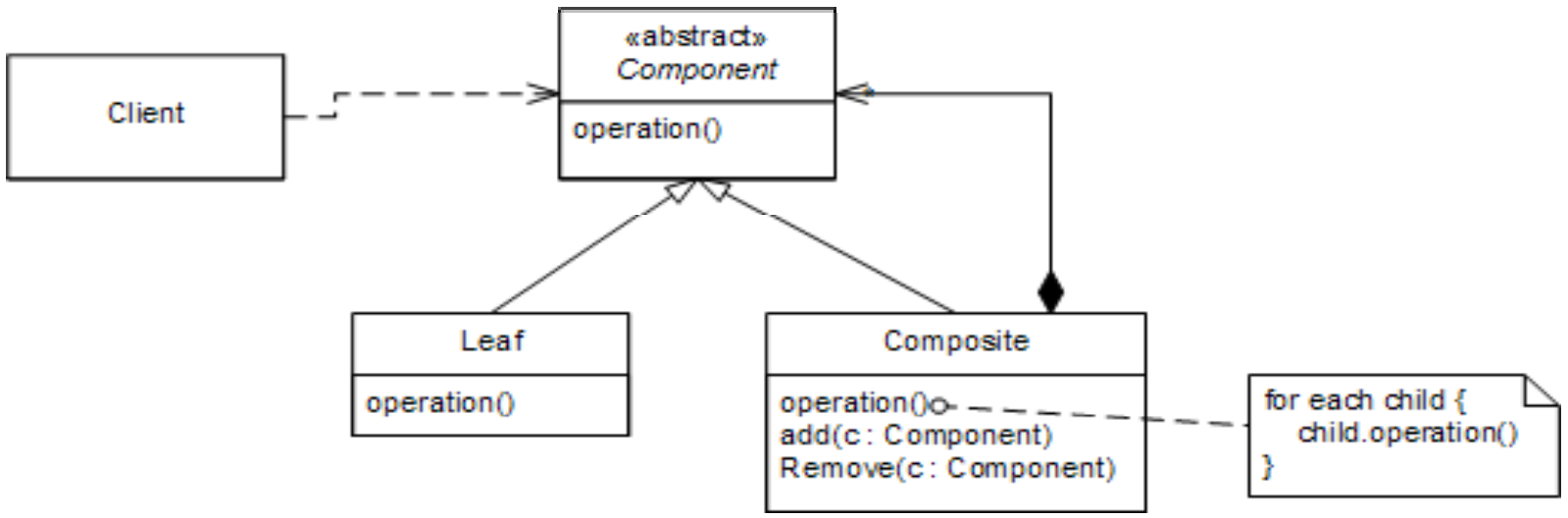
- Folder:
  - Za svaku akciju (delete, display, copy itd), postoji poseban tretman za fajlove i foldere.
- Explorer:
  - Svakim tipom objekata se manipuliše odvojeno.
- Skalabilnost:
  - Upravljanje većim brojem elemenata (diskovi, CD, USB…)

Composite (Sastav) - rešenje



Composite: Struktura

- ✧ Apstraktna osnovna klasa (Component) definiše jedinstveno ponašanje.
- ✧ Primitive i Composite klase su podklase.
- ✧ Composite definiše ponašanje komponenti koje imaju decu i čuva komponente decu.



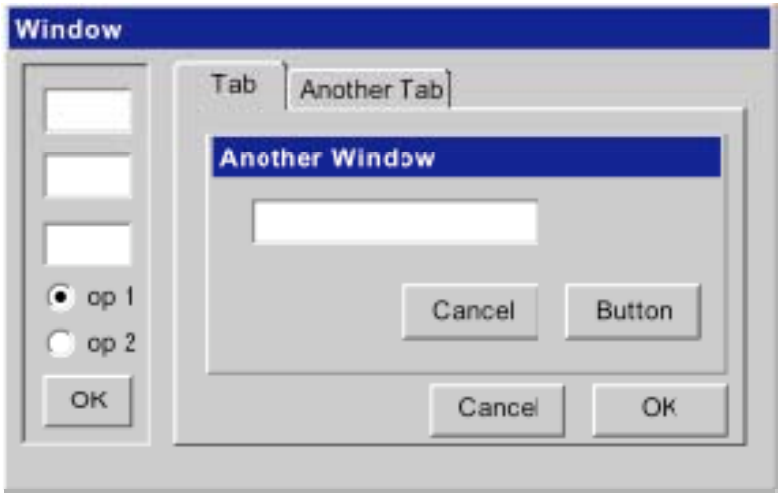
Ilustracija

Composite: Posledice

- Dobre
  - Olakšava dodavanje novih vrsta komponenti.
  - Pojednostavljuje posao klijentu koji obično ne znaju da li imaju posla sa listom ili sa složenom komponentom.
- Loše
  - Može da dovede do previše uopštenog dizajna, čime se teže postavljaju ograničenja za komponente sastava.

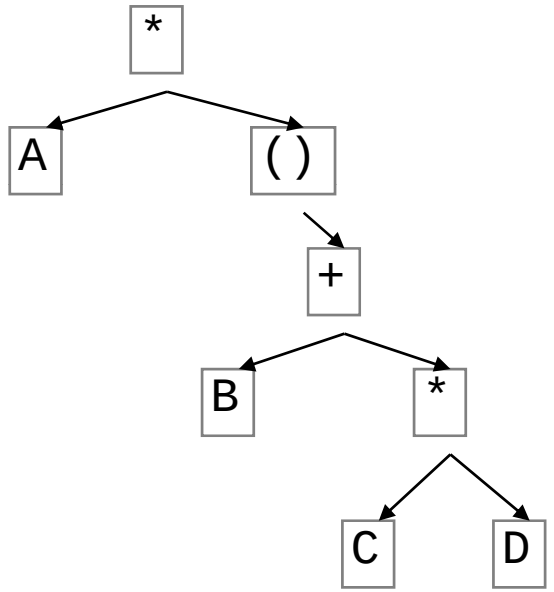
Composite: Primeri korišćenja

Grafički korisnički interfejs



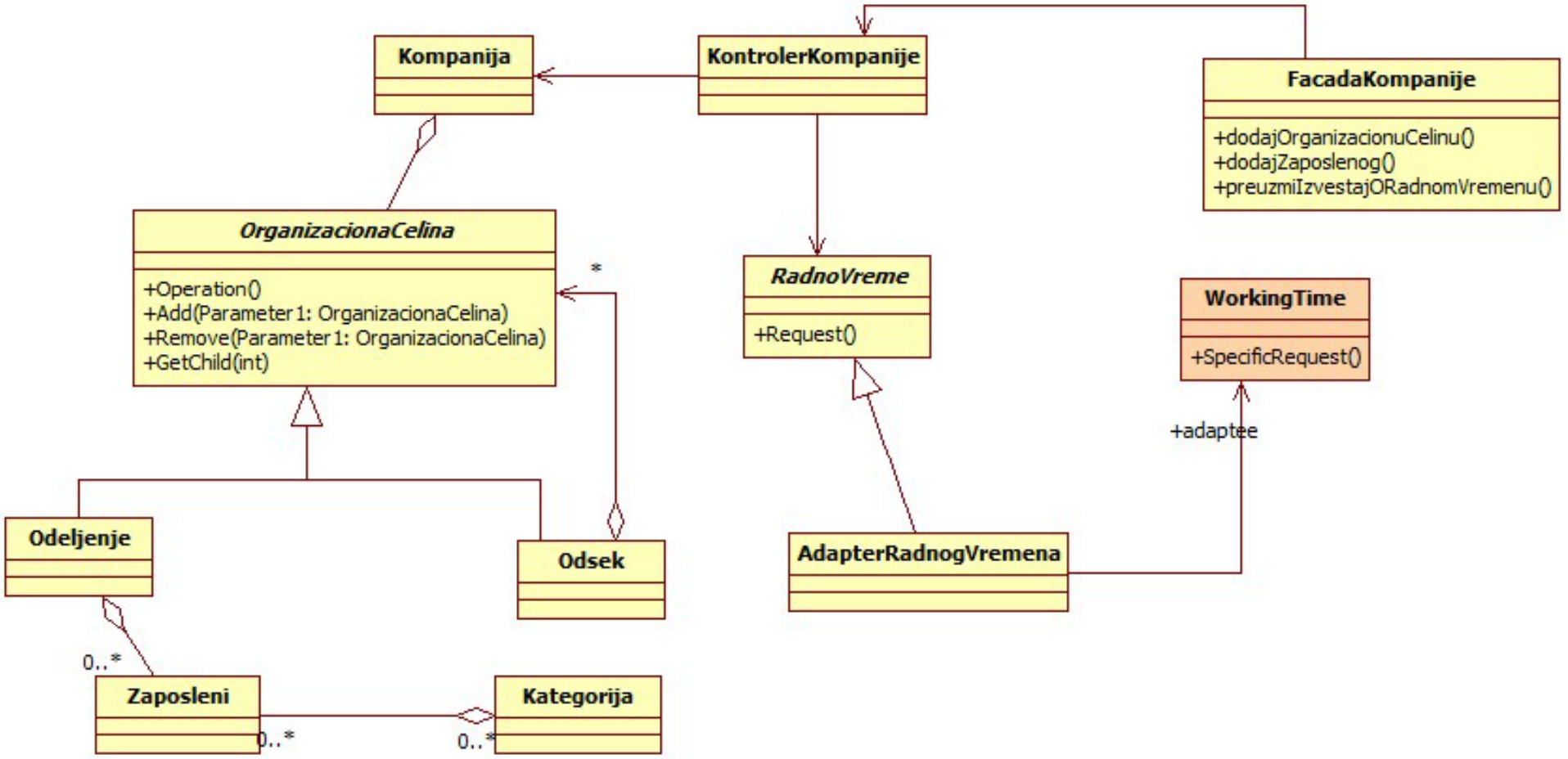
Aritmetički izrazi

A * (B + (C * D))



Kadrovski informacioni sistem

Predložiti model kadrovskog IS jedne kompanije koja vodi evidencije o svojim zaposlenima (detalje kao što su kategorije zaposlenih, podaci i slično definiše student). Kompanija je organizovana hijerarhijski po celinama (odseci i odeljenja), pri čemu se odseci mogu medjusobno ugnježdavati tako da odseci sadrže i druge odseke, dok su odeljenja najniže organizacione celine koje se ne mogu dalje deliti i mogu imati najviše 10 zaposlenih (Composite pattern). S obzirom da kompanija već poseduje IS za vođenje evidencije o radnom vremenu čije su funkcionalnosti dostupne preko klase WorkingTime a koja nije u potpunosti prilagođena zahtevima sadašnjeg informacionog sistema, potrebno je primenom Adapter paterna omogućiti integrisanje ova 2 sistema u jednu celinu, kako bi se o zaposlenima vodila evidencija o radu. Sistem treba da implementira funkcionalnosti obračuna zarade na osnovu evidencije o radu zaposlenih. Osim toga, potrebno je impementirati sistem na takav način da obezbedi zajednički interfejs (pristup svim funkcionalnostima) sa jedne tačke, kako bi se omogućilo jednostavno integrisanje sistema sa drugim sistemima koje kompanija planira da razvije (Facade patern).


Kompanija
KontrolerKompanije
FacadaKompanije
+dodajOrganizacionuCelinu()
+dodajZaposlenog()
+preuzmiIzvestajORadnomVremenu()
OrganizacionaCelina
+Operation()
+Add(Parameter1: OrganizacionaCelina)
+Remove(Parameter1: OrganizacionaCelina)
+GetChild(int)
*
RadnoVreme
+Request()
WorkingTime
+SpecificRequest()
+adaptee
Odeljenje
Odsek
AdapterRadnogVremena
0..*
Zaposleni
Kategorija
0..*
0..*

Softversko rešenje diskusionih foruma

Potrebno je dati predlog modela softvera za upravljanje diskusionim forumima. Softver treba da omogući kreiranje različitih kategorija koje mogu sadržati diskusione teme (discussion thread) ili druge pod-kategorije. U okviru jedne diskusione teme korisnici mogu poslati svoje poruke kao odgovor na temu. Na svaku poruku korisnici mogu slati komentare ili ocenjivati poruku ocenom od 1 do 5. Sistem treba da pruži korisnicima mogućnost unosa novih kategorija kao i započinjanja novih diskusionih tema. Za unos novih poruka kao odgovora na određenu temu, koristi se postojeći teks editor (klasa *TextEditor*) koja poseduje metodu *editorInitialisation()* i *saveContent()* koje nisu kompatibilne sa postojećom aplikacijom, te je potrebno prilagoditi ih potrebama diskusionog foruma.

U cilju obezbeđivanja mogućnosti korišćenja softverskog rešenja u drugim aplikacijama potrebno je obezbediti zajednički pristup svim osnovnim funkcionalnostima sistema.

Informacioni sistem svetskog prvenstva u fudbalu

Potrebno je dati predlog modela informacionog sistema u fudbalu. Potrebno je voditi evidenciju o reprezentacijama učesnicima takmičenja sa osnovnim atributima: šifra države, naziv države, kao i njenim igračima sa sledećim osnovnim atributima, broj na dresu, ime i prezime, pozicija, godište. Svaka reprezentacija je raspoređena u jednu i samo jednu kvalifikacionu grupu. Sve utakmice jedne kvalifikacione grupe se igraju u samo jednom gradu. Potrebno je modelovati utakmice, gde se tačno zna ko je u ulozi domaćina a ko u ulozi gosta, kao i koji je konačni rezultat utakmice. Osim toga, sistem omogućava registrovanje novinara ili TV reportera za praćenje svih događaja i značajnih podataka. Registracijom novinara u sistemu, sistem preuzima obavezu da ga obaveštava o svim aktuelnim promenama rezultata, početku i završetku utakmice, timovima, itd.