Projektni uzorci - strukturni (Structural design patterns-DP)

Dekorater (engl. *Decorator*)– strukturni objektni DP

Drugo ime:
Omotač (engl. *Wrapper*)

Namena:
–dinamički pridružuje dodatne odgovornosti (mogućnosti) nekom objektu
Primer: Funkcionalnost objekta se može proširiti statički nasleđivanjem (compile time). Međutim, često je neophodno dinamički dodati nove mogućnosti objektu tokom rada są objektom (runtime).

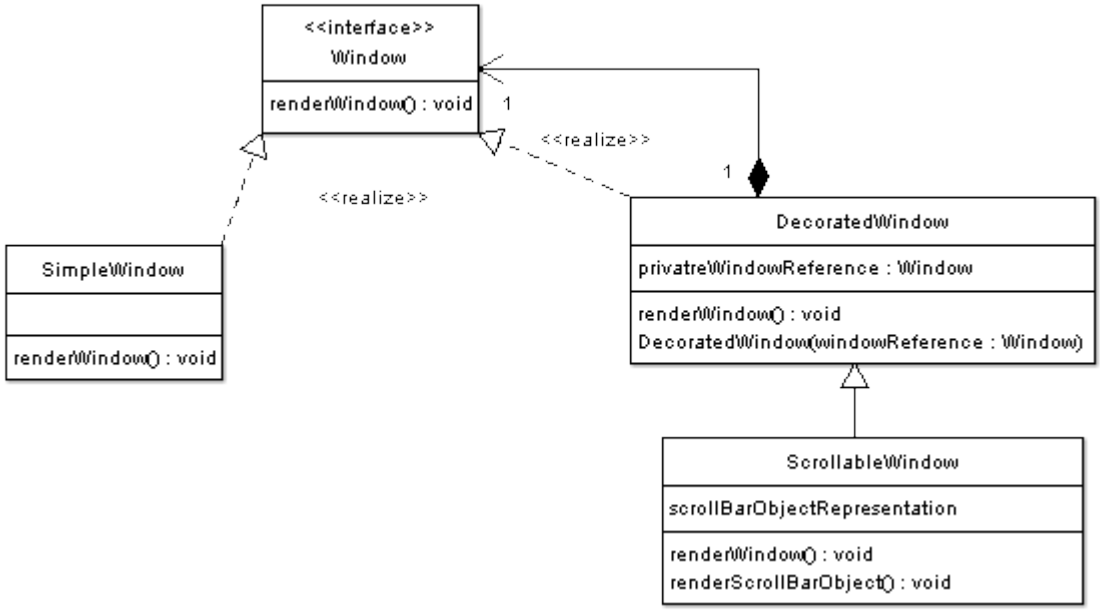
–predstavlja fleksibilnu alternativu mehanizmima nasleđivanja radi proširivanja funkcionalnosti
Primer rada sa grafičkim korisničkim interfejsom: Dodavanje novih funkcionalnosti grafičkom prozoru (npr. dodavanje klizača) zahteva nasleđivanje klase Window da bi se kreirala klasa `ScrollableWindow`, tj. da bi se kreirao objekat klase `ScrollableWindow`. Ali, time je nemoguće početi rad sa osnovnim prozorom i dodati mu funkcionalnost u vreme izvršavanja (runtime) kako bi postao prozor sa klizačem.

```
public class GUIDriver {
        public static void main(String[] args) {
                // kreiranje novog prozora
                Window window = new ConcreteWindow();
                window.renderWindow();

            // u nekom trenutku, veličina teksta može postati veća od prozora,
            // te se mora dodati klizač

                // dekoracija starog prozora
                window = new ScrollableWindow(window);

             //  sada objekat window ima dodatno svojstvo
                window.renderWindow();
        }
}
```



Primenljivost:

–za dodavanje odgovornosti pojedinačnim objektima na dinamički i transparentan način, tj. bez uticaja na druge objekte

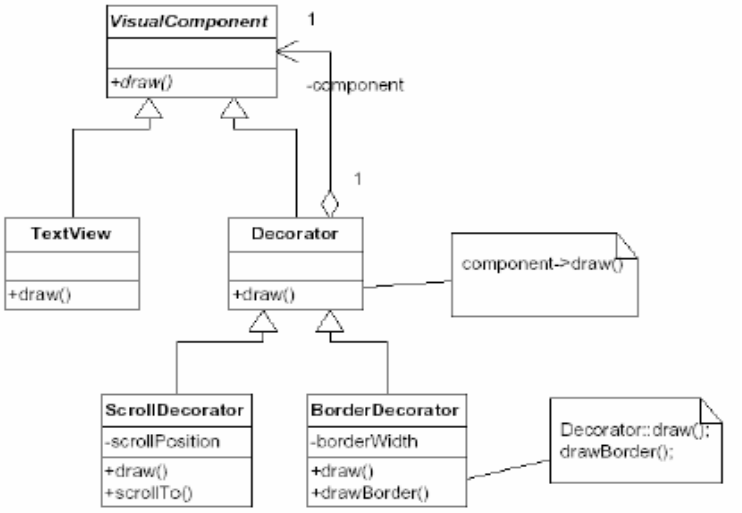
–kada proširenje pomoću podklasa (izvođenjem) nije praktično. Ponekad je moguće veliki broj nezavisnih proširenja, te bi podrška za svaku kombinaciju proizvela previše podklasa. Postoje i slučajevi kada je definicija klasa skrivena ili je na drugi način onemogućeno pravljenje podklasa.

Kratak opis problema

–alati za GUI podržavaju dodavanje klizača i okvira na razne komponente

Primer:

- neka objekat TextView prikazuje tekst u prozoru i nema okvir ni klizače za horizontalno ili vertikalno pomeranje sadržaja
- ako su potrebni klizači: dodati objekat klase ScrollDecorator (koji će dodati klizače).
- ako se želi i okvir: dodati i objekat klase BorderDecorator (koji crta okvir)



1. jedan način za dodavanje mogućnosti rada sa okvirom i klizačem– nasleđivanje

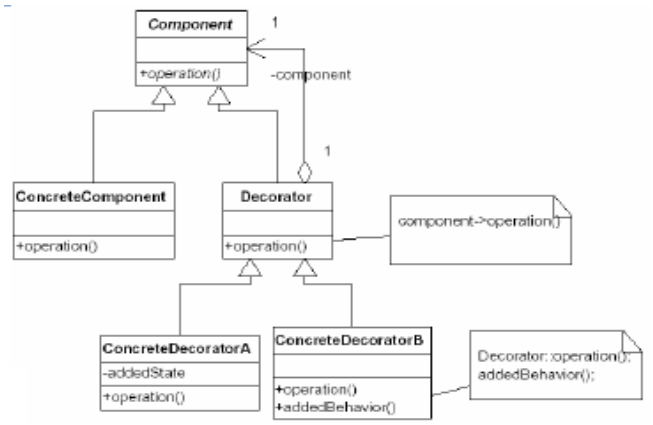
- nasleđivanje je nefleksibilno, jer bi se ukrasi birali statički (na primer: klijent ne može da bira kada da se iscrtava okvir)
- opasnost od eksplozije broja izvedenih klasa kada se kombinuju ukrasi

2. fleksibilniji pristup je da se komponenta obuhvati drugom komponentom

- **komponenta koja obuhvata osnovnu komponentu dodaje ukras** (okvir, klizače)
- proširenja mogu biti specifična stanja ili ponašanja
- **odgovornosti se dodaju pojedinom objektu, a ne celoj klasi**

3. komponenta koja obuhvata osnovnu se naziva **dekoraterom**
4. dekorater prosleđuje zahteve klijenta obuhvaćenoj komponenti i može da obavi dodatne akcije kao što je crtanje okvira ili dodavanje klizača za pomeranje
5. dekorater nasleđuje interfejs komponente koju ukrašava
6. prisustvo dekoratera je transparentno za klijente komponente. Transparentnost omogućava neograničen broj dodataka uvedenih posebnim dekoraterima

DIJAGRAM PRIKAZA



Učesnici:

Component klasa (u primeru VisualComponent)
definiše interfejs za objekte kojima se mogu dinamički dodavati mogućnosti
ConcreteComponent klasa (u primeru TextView)
definiše objekat za koji se mogu vezati dodatne mogućnosti
Decorator klasa (u primeru Decorator)
održava referencu na objekat Component i nasleđuje interfejs Component
ConcreteDecorator klasa (u primeru ScrollDecorator, BorderDecorator)
dodaje odgovornost objektu Component

Saradnja:

dekorater prosleđuje zahteve obuhvaćenom objektu (komponenti) i izvršava dodatne operacije pre i/ili posle prosleđivanja zahteva

Prednosti:

1. veća fleksibilnost od statičkog nasleđivanja (Decorator nudi fleksibilniji način dodavanja odgovornosti objektima nego što se postiže statičkim nasleđivanjem. Kada se koristi Decorator-i odgovornosti mogu da se uklanjaju i dodaju u vreme izvršavanja jednostavnim pripajanjem i rastavljanjem.)
2. izbegavanje bogatih parametrizovanih klasa visoko u hijerarhiji (Decorator nudi pristup kojim se složenost uvećava paralelno sa dodavanjem odgovornosti. Umesto pokušaja da se podrže sve moguće karakteristike pravljenjem složene prilagodljive klase, može se definisati jednostavna klasa kojoj će se pomoću *Decorator*-a postepeno dodavati funkcionalnosti.)
3. izbegavanje eksplozije podklasa koje kombinuju ukrase

Mane:

1. identitet dekoratera i dekorisanog objekta su različiti (Decorator služi kao transparentni omotač. Ali što se tiče identiteta objekata, dekorisani objekat nije identičan samoj komponenti. Prema tome ne bi se trebalo oslanjati na identitet objekta kada se koristi *Decorator*.)
2. objekti se ne razlikuju po klasi već po načinu povezivanja (potencijalan problem kod razumevanja i održavanja sistema zbog mnogo malih objekata. Ako se u projektu koristi *Decorator,* dobiće se sistem sastavljen od mnogo sličnih malih objekata, koji će se razlikovati samo po tome kako su povezani, a ne prema klasi ili vrednostima promenjivih. Takve sisteme lako prilagođavaju oni koji ih razumeju, ali sa druge strane teško ih je naučiti i analizirati greške.)

Bliski DP:

–*Decorator* zadržava interfejs, a *Adapter* menja interfejs
–*Composite* ima sličnu strukturu

- *Decorator* je po strukturi degenerisani *Composite*
- po nameni je *Decorator* sasvim različit, jer dodaje odgovornosti i nije namenjen grupisanju

–*Decorator* menja spoljašnjost objekta, a *Strategy* unutrašnjost –uzajamne alternative za promenu objekata

PRIMERI: Dekorator

Ilustracija.java
izlaz
KonkretnaKomponenta.Operacija()
KonkretniDekoratorA.Operacija()
KonkretniDekoratorB.Operacija()

RealanSlucaj.java
izlaz
Knjiga ------
Autor: Copic
Naslov: MagareceGodine
Broj kopija: 10

Video -----
Reziser: Spielberg
Naslov: Schindler's list
Broj kopija: 8
Vreme trajanja: 60

Film za pozajmicu:

Video -----
Reziser: Spielberg
Naslov: Schindler's list
Broj kopija: 6
Vreme trajanja: 60
Uzeo: Pera Peric
Uzeo: Mika Markovic

RealanSlucaj2.java
izlaz ?

Proksi (engl. ***Proxy***)– strukturni objektni DP

Drugo ime:
Surogat (engl.Surogate), Zamenik, Posrednik (za pristupanje drugom objektu radi ostvarivanje kontrole pristupa).

Namena:
–realizuje zamenu (surogat) drugog objekta koji:

- kontroliše pristup originalnom objektu
- odlaže punu cenu kreiranja i inicijalizacije originala do trenutka kada je ovaj zaista potreban

Primenljivost: uzorak treba koristiti god je za objekat potrebna složenija referenca od jednostavnog pokazivača:
Postoji više vrsta proksija prema primeni:

- **–udaljeni proksi (*remote proxy*)**

Ukoliko želimo da stvorimo iluziju klijentu da, pri pristupu udaljenom objektu, pristupa lokalnom objektu tada koristimo proxy koji delegira pozive udaljenim objektima (upotrebom tehnologija za distribuirane objekte - CORBA, DCOM, RMI i sl.).
Dakle, *Remote Proxy* obezbeđuje lokalnog predstavnika objekta koji se nalazi u drugom adresnom prostoru. Coplien ovu kategoriju proksija naziva *ambasadorom*.

- **–virtuelni proksi (*virtual proxy*)**

Ukoliko imamo objekat koji je *skup* za kreiranje sa stanovišta procesorskih/vremenskih zahteva, možemo da odložimo njegovo kreiranje za kasnije kada klijent zaista zatraži usluge servisa.
Dakle, *Virtual Proxy* kreira skupe objekte na zahtev, tj kešira dodatne informacije o subjektu da bi im se moglo naknadno iznova pristupiti (primer je ImageProxy)

- **– zaštitni proksi (*protection proxy*)**

Ukoliko želimo da obezbedimo bezbednosnu politiku pristupa objektima možemo da koristimo tzv. *Protective Proxy* koji proverava da li klijent ima pravo da izvrši operaciju koju namerava.
Dakle, *Protective Proxy* kontroliše pristup originalnom objektu. Zaštitni proksi treba da obezbedi različita prava pristupa.

- **–pametna referenca (*smart reference*)**

zamena za obični pokazivač, obavlja dodatne akcije prilikom pristupa objektu. Uobičajene su sledeće primene pametne reference:
1. brojanje referenci na pravi objekat da bi se on automatski oslobodio kada nema više referenci na njega.
2. učitavanje trajnog objekta u memoriju prilikom prvog referisanja
3. provera da li je objekat zaključan pre pristupanja da bi se obezbedilo da ga neki drugi objekat ne promeni.

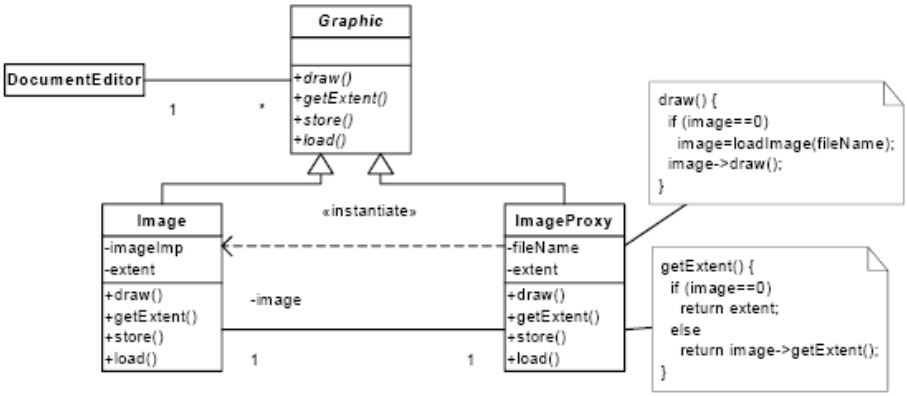
Kratak opis problema:

Razmotrimo poželjna svojstva za **editor dokumenata koji mogu da sadrže grafičke objekte u sebi**

- neki grafički objekti, na primer, velike rasterske slike su skupi za kreiranje, dok sa druge strane, otvaranje dokumenta treba da bude brzo
- pri otvaranju dokumenta, treba izbeći kreiranje svih slika koje dokument sadrži
- sliku treba otvarati na zahtev tek kad postaje stvarno potrebna, a činjenicu da se slika kreira tek na zahtev sakrivamo, da se ne bi komplikovao editor i ova optimizacija ne treba da utiče na kod za prikazivanje ili formatiranje dokumenta

Rešenje:

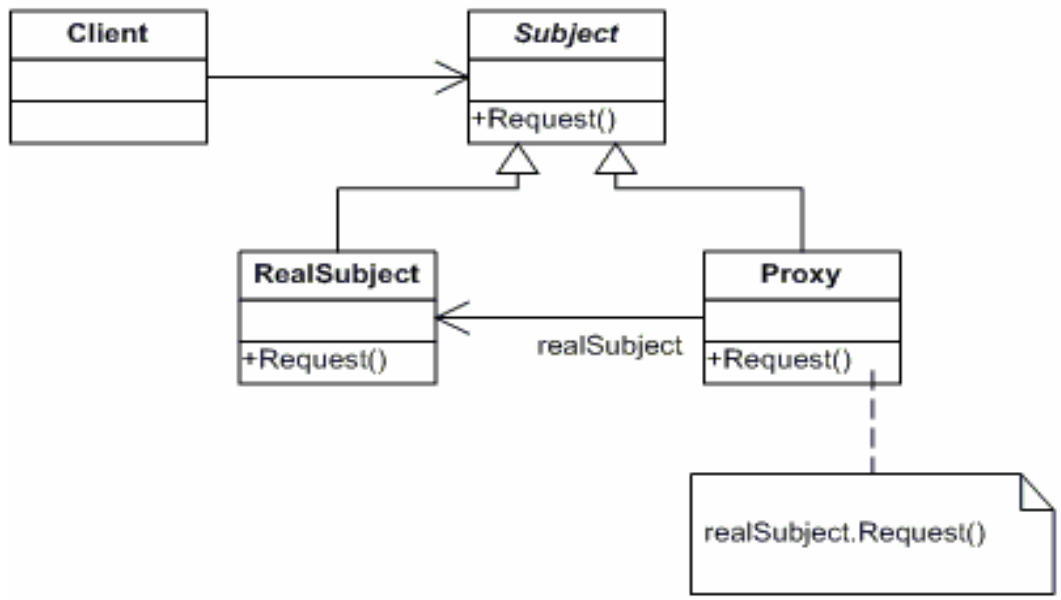
- u dokument stavljamo umesto realne slike poseban proksi-objekat koji je zamenjuje
- proksi se ponaša kao slika i vodi brigu o stvaranju objekta slike kad je baš on potreban
- u slučaju da su slike u posebnim fajlovima, proksi čuva ime fajla kao referencu na sliku
- proksi slike takođe čuva dimenzije slike
- svest o dimenzijama čini da proksi daje taj podatak formateru i bez učitavanja slike
- proksi slike stvara objekat slike tek kada editor od njega zahteva da se slika prikaže
- naredne zahteve proksi prosleđuje slici, tj. proksi čuva referencu na objekat slike



Editor dokumenata pristupa ugrađenim slikama kroz interfejs apstraktne klase Graphic

1. ImageProxy je klasa za slike koje se kreiraju na zahtev
2. ImageProxy održava ime fajla kao referencu na sliku koja je na disku
3. ime fajla se prosleđuje kao argument konstruktora ImageProxy
4. ImageProxy takođe čuva dimenzije slike i referencu na objekat stvarne slike
5. referenca na objekat stvarne slike nije validna dok se objekat slike ne kreira
6. operacija draw() proverava da li je objekat slike kreiran pre nego što mu prosledi zahtev
7. operacija getExtent() prosleđuje zahtev objektu slike, ako ovaj postoji
8. ako slika nije kreirana, getExtent() vraća sačuvane dimenzije slike

DIJAGRAM PRIKAZA



Učesnici:

RealSubject klasa (u našem primeru klasa Image)
realni objekat koji je reprezentovan proksijem

Subject klasa (u našem primeru klasa Graphic)
definiše zajednički interfejs za *RealSubject* i *Proxy*, da se *Proxy* može koristiti gde god se očekuje *RealSubject.*

Proxy klasa (u našem primeru klasa ImageProxy)

- čuva referencu za pristup stvarnom subjektu
- realizuje interfejs subjekta tako da može predstavljati zamenu za realan subjekat
- kontroliše pristup realnom subjektu; Proxy može biti odgovoran za njegovo kreiranje/uništavanje
- odgovornosti u zavisnosti od tipa proksija:

–*remote proxy* je odgovoran za slanje zahteva i argumenata u drugi adresni prostor
–*virtual proxy* može keširati dodatne informacije o realnom subjektu da odloži pristup
–*protection proxy* proverava da li pozivalac ima pristupnu dozvolu za dati zahtev

Saradnja:

Proxy prosleđuje zahteve RealSubject kada to odgovara, u zavisnosti od vrste proksija

Prednosti i mane:

1. *Proxy* unosi određen stepen indirekcije u pristupanje objektu. Dodatna indirekcija ima različite svrhe, u zavisnosti od vrste proksija:
• *Remote Proxy* može da sakrije činjenicu da se objekat nalazi u drugom adresnom prostoru.
• *Virtual Proxy* može da obavi optimizacije, kao što je pravljenje objekta na zahtev.
• *Protection Proxy* i pametne reference omogućavaju dodatne zadatke održavanja kada se pristupi objektu.

2. Proxy uvodi jedan nivo indirekcije u pristupu objektu
–optimizacija koju proksi može da sakrije od klijenta je *copy-on-write*

- kopiranje velikih i komplikovanih objekata može biti skupa operacija

- ako se kopija nikad ne modifikuje, nije neophodno platiti cenu
- korišćenjem proksija da odloži kopiranje, cena se plaća samo ako se objekat modifikuje
- zahtev za kopiranje rezultuje samo u inkrementiranju broja referenci na subjekat
- kada klijent zahteva operaciju koja modifikuje subjekat, tada ga proksi stvarno kopira
- tada proksi mora da dekrementira broj referenci na subjekat
- kada broj referenci padne na 0 –subjekat se uništava

Bliski DP:

–*Adapter* obezbeđuje različit interfejs objektu koji adaptira, dok *Proxy* obezbeđuje isti interfejs

–*Decorator*–može imati sličnu implementaciju kao *Proxy*, ali ima različitu namenu:

- dekorater dodaje jednu ili više odgovornosti objektu, a proksi kontroliše pristup objektu
- *protection proxy* može biti implementiran baš kao dekorater

PRIMERI: Zastupnik

Ilustracija.java

pozvan je RealSubject.Request()