21.01.2022.

Objektno-orijentisano programiranje 2

Podsetnik

Motivacija i osnovna ideja:

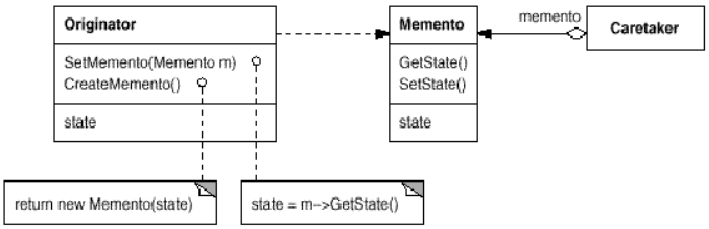
U mnogim aplikacijama potrebno je čuvati istoriju stanja nekog objekta. To najčešće činimo da bismo omogućili vraćanje stanja objekta na neko prethodno stanje. Kod razmatranja projektnog uzorka *Komanda* vraćanje prethodnog stanja realizovali smo bez čuvanja stanja samog objekta. Stanje smo vraćali tako što je svaka komanda „znala“ kako da poništi svoje dejstvo na objekat klase *Receiver.*

Pretpostavimo sada da želimo da omogućimo vraćanje stanja objekta tako što ćemo čuvati stanje objekta. Ovde pod stanjem podrazumevamo vrednosti svih relevantnih atributa datog objekta. Konkretnije, prilikom korišćenja objekta neke klase, zvaćemo je u skladu sa projektnim uzorkom koji razmatramo *Subjekt*, želimo da omogućimo da sačuvamo stanje tog objekta i da možda u nekom kasnijem trenutku vratimo stanje objekta na prethodno sačuvano stanje. Trivijalno rešenje je da svi atributi klase *Subjekt* budu javni (ili da imaju i setere i getere) i da „od spolja“ lako možemo da ih uzmemo i kasnije postavimo na stare vrednosti. Ovo nije dobra praksa, jer možda se među atributima nalaze i neki „osetljivi“ atributi kojima pristup ne bi trebalo da bude dostupan.

Rešenje koje nudi projektni uzorak *Podsetnik* je da stanje klase *Subjekt* (na engleskom se često naziva *Originator*) pamtimo u posebnom objektu koji nazivamo *Podsetnik* (engl. *Memento*). Subjekt je zadužen za kreiranje Podsetnika i Subjekt treba da omogući rekonstrukciju sopstevnog stanja na osnovu prosleđenog Podsetnika. Time omogućujemo čuvanje stanja, bez narušavanja inkapsulacije Subjekta. Obavezu da čuva podsetnik (jedan ili veći broj njih) i pruža pristup do njega (ili njih) ima klasa koju nazivamo *Čuvar* (engl. Caretaker). To je jedina funkcionalnost koju *Čuvar* po specifikaciji uzorka **mora** da obezbedi. U konkretnoj implementaciji, u zavisnosti od potreba, kôd se može organizovati tako da Čuvar ima pokazivač (ili referencu) na subjekta i da Čuvar ima metode *save()* i *restore()* kojim od Subjekta zahteva kreiranje Podsetnika koji će sačuvati i vraćanje prethodnog stanja Subjekta na osnovu sačuvanog Podsetnika, redom.

Napomena: korišćenje Podsetnika nije isključivo sa korišćenjem Komande. Klasa *Pokretač* može biti ujedno i *Čuvar* koji prilikom svakog izvršenja neke komande (automatski ili na zahtev) čuva *Podsetnik* koji omogućuje rekonstrukciju objekta *Primalac*. Naravno, moguće su kombinacije i sa drugim projektnim uzorcima. Na primer, stanje iz projektnog uzorka Stanje možemo čuvati kao podsetnik.

UML dijagram projektnog uzorka *Podsetnik* dat je na slici 1.



Slika 1: Projektni uzorak Podsetnik

Zadaci:

1. Implementirati opšti oblik projektnog uzorka *Podsetnik* u skladu sa UML dijagramom sa slike 1.
2. Implementirati klase Slika (ima RGB matrice piksela, a kreira se na osnovu putanje do fajla) i Dokument koja ima naslov, tekst i listu slika sa svojim pozicijama u dokumentu (pozicije čuvati kao brojeve). Omogućiti čuvanje stanja dokumenta pomoću *Podsetnika* i rekonstrukciju dokumenta na osnovu prosleđenog *Podsetnika*. Podsetnik treba da umesto samih slika čuva putanje do slika i njihovu poziciju.
3. Simulirati kretanje igrača u nekoj jednostavnoj igri po 2D mapi (matrici). Omogućiti čuvanje stanja igre. Kod igre se zna na kojoj je poziciji igrač i poznata je matrica polja nivoa na kome je igrač (recimo da je u pitanju kretanje kroz lavirint i da svako polje može biti slobodno, prepreka ili ciljno). Matrica nivoa učitva se iz fajla na osnovu rednog broja nivoa. Stanje igre može da se sačuva na osnovu rednog broja nivoa i imena i pozicije igrača. Stanja igre čuvaju se u fajlu i igra se može startovati na osnovu ranije snimljenog stanja.