28.12.2021.

Objektno-orijentisano programiranje 2

Posmatrač

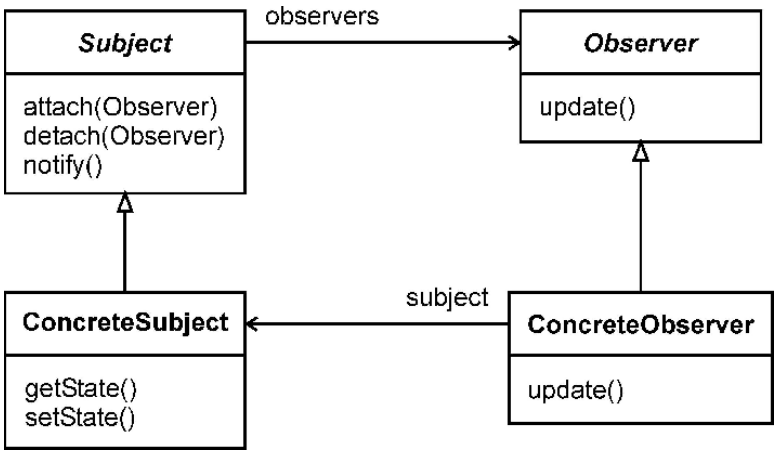
Motivacija i osnovna ideja:

Pretpostavimo da imamo objekat na čiju promenu stanja treba modifikovati ili izvršiti neku logiku nad drugim objektima. Drugi objekti mogu, ali ne moraju da budu istog tipa. Na primer, promena nekog objekta možda treba pokrenuti logiku za ponovno iscrtavanje aplikacije (ako imamo aplikaciju sa grafičkim interfejsom), upis u log fajl kao i slanje obaveštenja na mejl. Ponašanje koje želimo da postignemo je da imamo objekte koji "prate", "posmatraju" neki ciljni objekat čekajući na neku promenu i reaguju na datu promenu. Ili, gledano iz drugog ugla, kao da objekat može da saopšti svojim pratiocima, tj. posmatračima da je došlo do neke promene. Tipičan realni primer je kod kontrole pristupa kritičnoj sekciji u multithread aplikcijama, gde nakon što neka nit obavi određeni posao ili neka promenljiva promeni vrednost, treba obavestiti ostale niti o tome. Školski primeri su pratioci na društvenim mrežama koji dobiju obaveštenje kada objekat koji prate promeni stanje.

Projektni uzorak *posmatrač* (engl. *observer*) može se koristiti u ovakvim situacijama. Objekat na čije promene čekamo nazivamo *subjektom*, a subjekat vodi evidenciju o svojim *posmatračima* (može da ih dodaje i uklanja). Onda kada se subjekat promeni, treba da obavesti sve svoje posmarače (nekada se nazivaju i pratioci (engl. *subscribers*)) o tome na način da pozove određenu metodu nad svakim posmatračem. Posmatrači mogu biti objekti različitih klasa, ali potrebno je da imaju zajedničku roditeljsku klasu sa deklaracijom metode *ažuriraj* (engl. *update*) koju će pozivati subjekat kad god se načini neka promena njegovog stanja. Update kao argument mora da ima neki atribut od značaja za posmatrače. Često se kao atribut šalje pokazivač ili referenca na subjekta (najprostija i najuniverzalnija opcija).

U opštem slučaju može postojati veći broj različitih *subject* klasa (izvedenih iz iste roditeljske klase, naravno).

UML dijagram projektnog uzorka Posmatrač dat je na slici 1.



Slika 1: Projektni uzorak Posmatrač

Zadaci:

1. Implementirati opšti oblik projektnog uzorka Posmatrač u skladu sa UML dijagramom na slici 1.
2. Napisati klase Igrač i Igra za simulaciju neke video igre. Igra ima matricu polja i listu igrača. Na neko polje matrice može pasti prepreka. Igrač se nalazi na određenom polju. Igrač ima svoju verziju matrice polja i svako polje može biti neistraženo, čisto, zauzeto igračem ili prepreka. Kada na polje padne prepreka, svi igrači udaljeni jedno polje od datog polja treba da ažuriraju svoju veziju matrice. Postoji i brojač živih igrača (kao zasebna klasa). Kada god se doda novi igrač brojač treba povećati za 1, a ukoliko prepreka padne na igrača, datog igrača treba ukloniti iz igre (i smanjiti brojač).
3. Napisati klase JednostrukoPovezanaLista i Stek i omogućiti trenutno ažuriranje informacije o veličini, minimalnom i maksimalnom elementu i zbiru elemenata pomoću zasebnih “posmatrača”. Radi jednostavnosti, u ove strukture samo vršiti dodavanje elemenata.

Posrednik

Motivacija i osnovna ideja:

Osnovna ideja projektnog uzorka Posrednik je da komunikacija između srodnih objekata ne bude direktna. Ako objekat A treba da “pošalje poruku” objektu B ili možda većem broju objekata, nećemo opterećivati klasu objekta A time da mora da zna sa kojim sve objektima može da komunicira. Možda su “poruke” zahtevi koji zahtevaju obradu nad svim objektima koji učestvuju u komunikaciji. Na primer, zamislimo sistem za odlučivanje o tome koji avion treba da sleti na

koju pistu i u koje vreme. Ukoliko bi svaki avion odlučivao za sebe uz komunikaciju sa svakim drugim avionom pojedinačno, logika bi postala veoma kompleksna i sklona greškama.

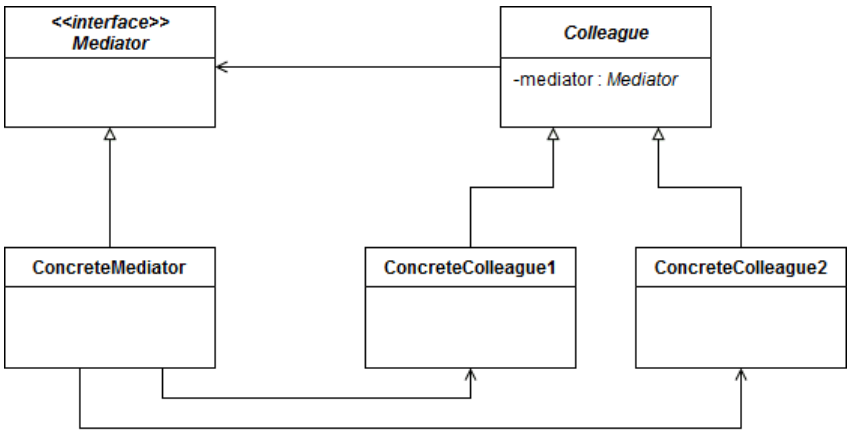
Zato uvodimo *posrednika* (engl. mediator) u komunikaciji. Objekti više ne šalju direktno poruke jedan drugome, već šalju poruku posredniku i od posrednika dobijaju odgovor. Na primeru aviona, avioni više neće komunicirati direktno međusobno, već će poruke slati kontrolnom tornju, koji upravlja komunikacijom.

U terminologiji projektnog uzorka Posrednik, objekti čiju komunikaciju treba organizovati nazivaju se *kolege*. U opštem slučaju imamo više različitih tipova konkretnih kolega izvedenih iz zajedničke apstraktne klase. Takođe, možemo imati više tipova posrednika (na primeru aviona, različiti tornjevi mogu primenjivati različitu logiku navođenja aviona).

Kolege moraju imati pokazivač (ili referencu) na posrednika kako bi mu poslali poruke, a posrednik takođe mora imati pokazivače na sve kolege čijom komunikacijom posreduje.

Napomena: u pitanju je veoma opšt projektni uzorak. Da li će kolege slati poruke namenjene isključivo jednom kolegi ili ćemo imati logiku u kojoj poruke utiču na sve kolege, to zavisi od prirode samog problema. Kakve će argumente imati metoda za komunikaciju sa posrednikom i da li će vraćati neki rezultat, takođe zavisi od konkretnog problema.

Učesnici projektnog uzorka Posrednik i njihova komunikacija dati su UML dijagramom prikazanim na slici 2.



Slika 2: Projektni uzorak Posrednik

Zadaci:

4. Implementirati klase Račun, DinarskiRačun, DevizniRačun, IBanka, BankaA, BankaB. Banke imaju listu računa, a imaju i listu blokiranih računa. Sa računa se, posredstvom banke, može zatražiti transakcija na neki drugi račun. Krus evra je 119/117 (prodajni/kupovni). Kada se zatraži transakcija sa računa, dobija se povratna informacija

o tome koliko zaista novca treba skinuti sa računa. Naime, konkretne banke računaju različite provizije za transakcije. Rezultat 0 znači da transakcija nije uspela.

5. Modifikovati zadatak 2 tako da bude moguće i pomeranje igrača za jedno polje. Kada se igrač pomeri, treba da o tome obavesti posrednika, a posrednik zatim obaveštava o tome sve ostale igrače.