Dizajn paterni ponašanja

Strategy patern

Strategy patern izdvaja algoritam iz matične klase i uključuje ga u posebne klase. Omogućava fleksibilan izbor algoritama bez izmjene klijentskog koda. Klijent bira odgovarajuću strategiju, a algoritmi su enkapsulirani i implementirani kroz zajednički interfejs. Prednosti uključuju zamjenu nasljeđivanja kompozicijom, dok su nedostaci potreba za dodatnim klasama.

State patern

State patern omogućava objektima da mijenjaju ponašanje u zavisnosti od svog unutrašnjeg stanja. Svako stanje je predstavljeno kao posebna klasa. Klasa konteksta delegira ponašanje trenutnom stanju. Prednosti uključuju pojednostavljenje koda i lakše održavanje, dok je nedostatak moguća prekompleksnost ako postoji mali broj stanja.

Template Method patern

Template Method patern definiše kostur algoritma u nadklasi, a prepušta podklasama da implementiraju specifične korake. Pruža mogućnost ponovne upotrebe koda i prilagodbu dijelova algoritma. Povezan je sa nasljeđivanjem i podržava tzv. Hollywood princip: 'Ne zovi nas, mi ćemo tebe zvati'.

Observer patern

Observer patern uspostavlja jednosmjernu zavisnost između objekata tako da promjena stanja u jednom objektu automatski obavještava sve njegove posmatrače. Koristan je za implementaciju događajnog sistema. Prednosti su skalabilnost i odvojenost subjekta i posmatrača, a nedostatak je nepredvidiv redoslijed notifikacija.

Iterator patern

Iterator patern omogućava sekvencijalni pristup elementima kolekcije bez otkrivanja njene interne strukture. Podržava više istovremenih iteratora nad istom kolekcijom. Klijent koristi zajednički interfejs, bez potrebe da zna konkretni tip kolekcije.

Command patern

Command patern enkapsulira zahtjev kao objekat, čime omogućava parametarsko pokretanje operacija, kao i mogućnosti undo/redo, red čekanja i zabilježavanje. Razdvaja pošiljaoca od izvršioca akcije, a koristi se gdje je potrebno odvojeno upravljanje operacijama.