Paterni ponašanja

Za "Pixel Vrtić", koji uključuje dinamičke procese kao što su registracija prisustva, obavještenja, analiza aktivnosti i upravljanje korisničkim profilima, upotreba paterna ponašanja omogućava modularizaciju funkcionalnosti, povećava ponovnu iskoristivost koda i olakšava testiranje.

Strategy pattern

Izdvaja algoritme iz matične klase i uključuje ga u posebne klase, omogućava kreiranje porodice algoritama i zamjenu algoritama bez potrebe za izmjenom klijentskog koda. Za naš projekat moguće je izdvojiti algoritam za izračunavanje naknade, te u ovisnosti kakve usluge koristi korisnik upotrijebiti različitu vrstu algoritma za obračun.

State pattern

Mijenja način ponašanja objekta na osnovu trenutnog stanja, odnosno ponaša se kao da mijenja klasu tokom izvršavanja programa. Na primjer, dijete može da bude u više različitih stanja na osnovu toga da li je prisutan, odsutan ili bolestan. Na osnovu tog stanja, vrši se obrada prisustva ili se po mogučnosti mogu slati neke specijalne poruke i obavještenja roditeljima.

Template Method pattern

Omogućava algoritmima da izdvoje neke korake u podklase, dok se sama struktura algoritma nalazi u nadklasi. Moguće je razdvojiti algoritam za kreiranje izvještaja gdje se koraci dobavljanja podataka, formatiranja podataka i ispisa podataka stavljaju u podklase. Date nadklase mogu biti različiti tipovi izvještaja u odnosu na šta se odnosi sam izvještaj ili u zavisnosti kojem korisniku se šalje izvještaj.

Chain of Responsibility pattern

Predstavlja listu objekata, ukoliko objekat ne može da odgovori proslijeđuje se narednom u nizu sve dok se zahtijev ne može uspješno izvršiti. Na primjer, ako roditelj šalje upit o promjeni termina neke aktivnosti, zahtjev se šalje prvo vaspitaču koji je odgovoran za taj termin, ako on nije u mogučnosti da odgovori, zahtjev se šalje narednom vaspitaču i proces se nastavlja sve dok neko ne može da odgovori ili dok se ne dođe do kraja niza.

Command pattern

Zahtjev pretvara u samostalni objekat. Na taj način se mogu jednostavno proslijediti, staviti u red čekanja ili zabilježiti zahtjev i podržati operacije poput undo/redo. Kao što smo za chain of responsibility slali zahtjev i čekali odgovor za promjenu termina, ako niko u datom trenutku nije u mogučnosti da odgovori, zahtjev se sačuva. Roditelj onda može da promjeni sadržaj početnog zahtjeva i neko od vaspitača koji bude u mogučnosti može da odogovori na datu poruku.

Iterator pattern

Omogućava pristup elementima kolekcije sekvencijalno bez poznavanja interne strukture kolekcije. U našem slučaju imamo dosta kolekcija podataka, kao što su liste djece, roditelja, vaspitača, aktivnosi ili obavještenja. Možemo kreirati paterne za bilo koju od ovih lista podataka ako je potrebno. Na primjer za pregled sve djece u određenoj grupi ili za pregled svih aktivnosti koje su zadane od strane nekog vaspitača.

Mediator pattern

Enkapsulira protokol za komunikaciju među objektima dozvoljavajući da objekti komuniciraju bez međusobnog poznavanje interne strukture objekta, odnosno da nema potrebe stvaranja direktnih veza među objektima. Obzirom na veliki broj komponenti našeg sistema i potrebom za komunikaciju između roditelja i vaspitača, roditelja i obavještenja i drugih, možemo uvesti medijatore za komunikaciju između navedenih pojedinačnih klasa ili jedan medijator koji bi bio odgovoran za komunikaciju između svih klasa sistema.

Observer pattern

Uspostavlja relaciju između objekata takvu da kad se stanje jednog objekta promijeni svi vezani objekti dobiju informaciju. Moguće je koristiti veliki broj observera o našem sistemu. Ako roditelj označi da dijete sutra neće biti prisutno, svi vaspitači zaduženi za tu grupu koje dijete pohađa će dobiti obavijest o tome, ili ako se odgodi neka aktivnost svi roditelji čija djeca su trebala prisustvovati toj aktivnosti će znati da se odgodilo.

Visitor pattern

Definira i izvršava nove operacije nad elementima postojeće strukture ne mijenjajući samu strukturu. Ima generalno široku upotrebu, na primjer podatke o djeci možemo da exportujem u različitim formama u zavisnoti koju obradu je potrebno izvršiti, nove metode za kreiranje različitih vrsta izvještaja i druge. Najveća koriste ke dodavanje kompletne nove funkcionalnosti koja zahtjeva veliki broj novih metoda kao što je dodavanje sekcija različitih vrsta za sport, ples ili crtanje.

Interpreter pattern

Podržava interpretaciju instrukcija napisanih za određenu upotrebu. Trenutno nema neku potrebu dodavati, ali ako bi naš sistem koristio neki speijalan način za filtriranje djece ili filtriranje aktivnosti, onda bi koristili interpretere da bi mogli taj tekst ili pravila koristiti u kodu.

Memento pattern

Omogućava spašavanje internog stanja nekog objekta van sistema i njegovo ponovno vraćanje. Na primjer ako napravimo neke promjene o podacima dijeteta ili roditelja koji su pogrešni, onda možemo da vratimo stare verzije podataka koji su bili ispravni. Na taj način imamo backup bitnih podataka.