Kreacijski paterni

1. Factory Method Pattern

Factory Method pattern služi za kreiranje objekata bez specifikovanja tačno koje klase će biti instancijane. Ovaj pattern definiše interfejs za kreiranje objekata, ali dozvoljava podklasama da odluče koja klasa će biti instancijana. Na taj način se omogućava kreiranje objekata kroz zajednički interfejs, čime se povećava fleksibilnost sistema.

U sistemu jahačkog kluba može se implementirati Factory Method pattern kroz klasu UserFactory, koja omogućava kreiranje različitih tipova korisnika (Guest, Član, Trener, Admin) na osnovu kategorije. Umjesto da se objekti kreiraju direktno pomoću konstruktora, koristi se factory metoda koja na osnovu proslijeđene kategorije određuje koji tip korisnika treba kreirati. Ovo omogućava lako dodavanje novih tipova korisnika u budućnosti bez mijenjanja postojećeg koda koji poziva factory metodu.

2. Builder Pattern

Builder pattern služi za kreiranje složenih objekata korak po korak. Omogućava kreiranje različitih reprezentacija objekta koristeći isti proces konstrukcije. Ovaj pattern je posebno koristan kada objekat ima mnogo opcionalnih parametara ili kada kreiranje objekta zahtijeva složenu logiku.

U sistem jahačkog kluba može se dodati Builder pattern za kreiranje Trening objekata. Trening ima mnoge opcione parametre kao što su naziv, datum, trener, lista konja, maksimalan broj članova i nivo treninga. TreningBuilder omogućava postupno kreiranje trening objekta sa validacijom na svakom koraku, čime se osigurava da se kreira valjan trening objekat.

3. Singleton Pattern

Singleton pattern osigurava da klasa ima samo jednu instancu i pruža globalni pristup toj instanci. Ovaj pattern je koristan kada je potrebno kontrolisati pristup dijeljenim resursima ili kada treba postojati samo jedan objekat određene klase u cijelom sistemu.

U sistemu jahačkog kluba, Singleton pattern se može efikasno primijeniti na klasu Logger, koja je zadužena za bilježenje važnih događaja tokom rada aplikacije — kao što su greške, prijave korisnika, rezervacije termina, promjene u bazi podataka itd. Budući da više komponenti sistema može istovremeno pokušati zapisati logove, korištenjem Singleton patterna obezbjeđuje se da svi ti zapisi idu kroz jednu jedinu instancu Logger klase, čime se izbjegava dupliranje, gubitak ili konflikt prilikom pisanja logova. Ovo omogućava dosljedno i centralizovano praćenje svih aktivnosti u aplikaciji.

4. Prototype Pattern

Prototype pattern služi za kreiranje objekata kloniranjem postojećih instanci umjesto kreiranja novih objekata od početka. Ovaj pattern je koristan kada je kreiranje objekta skupo ili kompleksno, a potrebno je kreirati mnoge slične objekte.

U sistem jahačkog kluba može se implementiran Prototype pattern za kreiranje template treninga. Trener može kreirati osnovni template treninga sa određenim postavkama, a zatim klonirati taj template i modificirati ga za različite termine ili nivoe, bez potrebe da svaki put unosi sve podatke od početka.

5. Abstract Factory Pattern

Abstract Factory pattern pruža interfejs za kreiranje familija povezanih ili zavisnih objekata bez specifikovanja njihovih konkretnih klasa. Ovaj pattern je koristan kada sistem treba biti nezavisan od načina kreiranja, kompozicije i reprezentacije objekata koji ga čine.

U sistem jahačkog kluba može se implementirati Abstract Factory pattern za kreiranje različitih tipova treninga i pratećih komponenti. Na primjer, možemo imati factory za kreiranje osnovnih treninga (za početnike) i naprednih treninga (za iskusne jahače), gdje svaki factory kreira odgovarajuće objekte (trening, vježbe, konje).