Kreacijski pattern

1. Singleton pattern

Singleton osigurava da postoji samo jedna instanca klase u cijelom sistemu.
Koristi se kada je potreban centralizovani pristup jednom objektu, npr. konfiguracija ili repertoar filmova.
U našem dijagramu koristimo Prototype pattern kako bi omogućile kloniranje objekta Karta preko interfejsa IPrototip, koji definiše metodu kloniraj(). Klasa KartaPrototip implementira ovaj interfejs, čime omogućava stvaranje kopija karata bez direktnog instanciranja, što je korisno za brzu i fleksibilnu izradu rezervacija.

2. Prototype pattern

Prototype omogućava kopiranje postojećih objekata, umjesto kreiranja novih kroz konstruktore.
Korisno je kada je kreiranje objekta skupo ili kompleksno, pa se koristi šablon (prototip).
U našem dijagramu koristimo Singleton pattern kroz klasu RepertoarSingleton, koja osigurava da postoji samo jedna instanca repertoara filmova pomoću metode getInstance(). Time omogućavamo centralizovano upravljanje listom filmova u sistemu, bez dupliciranja podataka.

3. Factory method pattern

Factory Method delegira kreiranje objekata podklasama pomoću metode koja vraća interfejs ili apstraktnu klasu. Klasa ne zna tačan tip objekta koji se kreira – to rješava podklasa kroz override metode.
Factory method bi se koristio u klasi Rezervacija za kreiranje objekta Cijena, tako da se logika za izračunavanje konačne cijene (sa ili bez popusta) centralizira u jednoj metodi. Umjesto da se Cijena direktno instancira vani, koristimo metodu create() koja vraća prilagođeni objekat Rezervacija.

4. Abstract factory pattern

Abstract Factory omogućava kreiranje grupa povezanih objekata bez direktnog navođenja njihovih konkretnih klasa. Koristi se kada postoji više porodica proizvoda i treba ih međusobno zamijeniti.
Abstract factory bi se koristio za kreiranje različitih tipova karata i sjedala kao jedinstvenih paketa – npr. VIP fabrika kreira VIP sjedalo i odgovarajuću kartu, dok Regular fabrika kreira regularne verzije. Na taj način možemo generisati povezane objekte bez direktnog navođenja njihovih konkretnih klasa.

5. Builder pattern

Builder razdvaja proces konstrukcije objekta od njegove reprezentacije.
Koristi se za kreiranje složenih objekata s mnogo parametara, gdje je konstruktor nepraktičan.
Klijent koristi "Builder" objekat da postepeno gradi finalni proizvod kroz lančane metode.
Builder pattern bi bio koristan za kreiranje objekta Projekcija, koji sadrži više atributa poput datuma, vremena, filma i dvorane. Umjesto kompleksnih konstruktora, koristimo fluentni pristup (setFilmId(), setDatum()...) kako bi postepeno gradio objekat.