**PATERNI**

* MOGUĆNOST IMPLEMENTACIJE U PROJEKTU „AV-MAU AZIL“ -

**Proxy pattern**

Namjena Proxy paterna je da omogući pristup i kontrolu pristupa stvarnim objektima. Proxy je obično mali javni surogat objekat koji predstavlja kompleksni objekat čija se aktivacija postiže na osnovu postavljenih pravila. Proxy patern rješava probleme kada se objekt ne može instancirati direktno (npr. zbog restrikcije pristupa). Struktura Proxy patterna je sastavljena od klasa:

1. Subject (ISubject) zajednički interfejs za realne/stvarne subjekte i proksije- surogate (proxies) koji omogućava da se oni koriste naizmjenično.

2. RealSubject je glavni objekat kojeg “predstavlja” proxy.

3. Proxy - implementira isti interfejs kao RealSubject tako da se Proxy može koristiti umjesto RealSubject objekta. Proxy vrši kontrolu pristupa RealSubject objektu - može kreirati i brisati taj objekat.

**PRIMJER UPOTREBE U PROJEKTU „AV-MAU AZIL“:** Ukoliko bi u budućnosti u Azilu bio razvijan sistem naplate karticom (na primjer za kupovinu hrane ili drugih rekvizita za ljubimce), svaki put kada se vrši naplata sa kartice, bio bi korišten proxy pattern. Kartica korisnika je ustvari proxy reprezentacija bankovnog računa. Proxy omogućuje naplatu sa kartice klijenta bez pristupa svim podacima računa korisnika (koji je u pozadini, i koji je implementiran „moćnijom“ klasom).

**Decorator pattern**

* Osnovna namjena Decorator paterna je da omogući dinamičko dodavanje novih elemenata i ponašanja (funkcionalnosti) postojećim objektima. Može se naprimjer koristi za implementaciju različitih kompresija videa, simultano prevođenje
* Decorator pattern čine sljedeće klase:

1. Component – Originalna klasa (koja sadrži interfejs koji se može mijenjati ili mu se mogu dinamički dodati operacije)

2. IComponent – interfejs koji identificira klase objekata koji trebaju biti dekorisani  
3. Decorator – klasa koja odgovara IComponent interfejsu i implementira dinamički prošireni interfejs

**Facade pattern**

* Facade patern se koristi kada sistem ima više identificiranih podsistema (subsystems) pri čemu su apstrakcije i implementacije podsistema usko povezane
* Osnovna namjena Facade paterna je da osigura više pogleda visokog nivoa na podsisteme (implementacija podsistema skrivena od korisnika)
* Može se više fasada postaviti oko postojećeg skupa podsistema i na taj način formirati više prilagođenih pogleda na sistem.
* Struktura Facade patterna sačinjena je od sljedećih klasa:

1. Facade – definira i implementira jedinstven interfejs za skup operacija nekog podsistema  
2. SubsystemClassN – definira i implementira N-ti interfejs u skupu interfejsa nekog podsistema

* Implementacija podsistema je u okviru odvojenih imenovanih prostora (ili biblioteka), u  
  kojima se sa specifikatorima pristupa može postaviti željena vidljivost klase.

**Singleton patern (unikat)**