**Singleton**

Budući da je potrebno da postoji samo jedna instancu ronilačkog kluba tijekom izvršavanja programa tako klasa *DivingClubSingleton* predstavlja uzorak *singleton*. Sukladno navedenom postoji samo jedna instanca ronilačkog kluba koju možemo dohvatiti preko metode definirane u istoj. Također, uzorak *singleton* je primijenjen i unutar Flyweight uzorka gdje je *FlyweightFactory* klasa *singleton*. U ovome slučaju uzorak *Flyweight* se mogao izvesti i bez korištenja uzorka *singleton*, ali unatoč tome htio sam osigurati da se pojavljuje samo jedna instanca *FlyweightFactory* klase tijekom izvršavanja programa te da se preko te jedinstvene instance izvršavaju metode definirane u istoj.

**Factory Method**

Uzorak Factory Method je korišten u aplikaciji za kreiranje objekta pojedinog algoritma. Sama klasa ne može predvidjeti klasu za objekte koje mora kreirati već se kreira klasa ovisno o unesenom argumentu prilikom pokretanja programa te je radi toga korišten navedeni uzorak. Postoje tri konkretna proizvoda koji nasljeđuju abstraktnu klasu koja definira samo jednu metodu koju trebaju implementirati podklase. Korištenjem ovoga uzorka podklasama dajemo mogućnost da odluče koju konkretnu klasu treba instancirati.

**Observer**

Nakon svakog urona ronilački klub jer dužan obavještavati agenciju/federaciju u kojoj pripada ronilac te također treba obavještavati nacionalni ronilački savez o pojedinom uronu svakog ronilaca pri čemu je važno pretpostaviti kako nisu unaprijed poznate sve agencije /federacije/institucije koje treba obavijestiti. Ovo predstavlja idealnu situaciju za implementaciju uzorka *observer* iz razloga što postoji kada se dogodi promjena stanja odnosno kreira se uron tada je ronilački klub dužan obavijestiti sve institucije koje će se onda automatski ažurirati. Također, uzorkom *observer* je moguće obavijestiti proizvoljni broj agencija/institucija što je i bio zahtjev u zadatku. Na ovaj način ronilački klub obavještava agencije i institucije, ali se ne veže čvrsto za iste.

**Flyweight**

U svrhu ograničavanja stvaranja velikog broja objekata klase *Certificate* i *Federation* iskorišten je uzorak *flyweight.* Navedenim uzorkom rješavamo problem pojave ronioca sa već postojećim certifikatom ili agencijom. Konkretna klasa *FlyweightFactory* se brine o tome da li objekt već postoji te ako postoji vraća isti, inače kreira novi i njega vraća. Na ovaj način se smanjuju troškovi spremanja, a također sam na ovaj način ograničio broj objekata klase *Federation* koje predstavljaju *observere.*

**Facade**

Klasa *DiverApplication* komunicira sa svim ostalim klasama u sustavu. Na ovaj način klijent šalje zahtjeve preko objekta navedene klase. *Facade* uzorak je iskorišten u svrhu zaštite komponenti sustava gdje je klijentu dozvoljeno samo izvršavanje glavne metode *DiverApplication* klase.

**Visitor (dodatna funkcionalnost)**

Nova funkcionalnost obuhvaća sigurnosnu provjeru rizika urona kod svih agencija/federacija i institucija tako da se traži onaj uron koji je bio najmanje siguran. Na temelju ovog izvještaja se mogu ubuduće planirati uroni koji će biti sigurniji. Na ovaj način definiramo operaciju pronalaska najmanje sigurnog urona na objektima *InstitutionAbstract* klase koja zapravo predstavlja agencije/federacije i institucije.

**Promjene u odnosu na rješenje prve zadaće**

Promjene u odnosu na rješenje prve zadaće ne postoje. Uzorci *singleton* i *factory method* koji su korišteni u prvoj zadaći isto su iskorišteni i u drugoj zadaći. Budući da nije bilo promjena strukture u odnosu na prvu zadaću, rješenje ove zadaće se samo nadograđivalo na završno stanje prve zadaće.

