**Singleton**

Budući da je potrebno da postoji samo jedna instancu klase međuspremnika tijekom izvršavanja programa tako klasa *Cache* predstavlja uzorak *singleton*. Sukladno navedenom postoji samo jedna instanca međuspremnika koju možemo dohvatiti preko metode definirane u istoj.

**Flyweight**

U svrhu ograničavanja stvaranja velikog broja objekata klase *Certificate* iskorišten je uzorak *flyweight.* Navedenim uzorkom rješavamo problem pojave ronioca sa već postojećim certifikatom ili agencijom. Konkretna klasa *CertificateFlyweightFactory* 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.

**Composite**

Zbog načina strukture kategorija i podkategorija idealan uzorak za navedenu situaciju se nameće uzorak *composite.* Pomoću uzorka *composite* sastavljamo objekte u strukturu stabla tako da predstavljaju hijerarhiju dio-cjelina gdje listove predstavlja konkretna oprema, a grane su kompozitni/složeni elementi odnosno kategorije i podkategorije.

**State**

Uzorak *State* sam iskoristio za mijenjanje ponašanja odnosno stanja objekta koji je zadužen za prikaz (*View*). Na ovaj se način ovisno o stanju View-u pridružuje konkretni *controller.* Kada se korisnik nalazi u dijelu sa popisom ronioca tada ta situacija predstavlja jedno stanje, a kada se nalazi na popisu opreme tada je u drugom stanju i drugi *controller* je povezan na *View.*

**Memento**

Budući da je u programu bilo potrebno implementirati mogućnost vraćanja na inicijalno spremljene ronioce tako da se unese „V“ iskoristio sam uzorak *Memento* u tu svrhu. Pomoću navedenog uzorka lako spremim stanja liste ronioca te se na zahtjev korisnika moguće vratiti na neko od početnih stanja objekta. U konkretnome slučaju potrebno je čuvati samo dva stanja liste ronioca tako da je klasa *Caretaker* zadužena za čuvanje inicijalnog stanja liste kako bi se na nju mogli kasnije vratiti ukoliko je potrebno.

**MVC**

Za rad s korisničkim sučeljem bilo je potrebno koristiti MVC uzorak. MVC dijeli aplikaciju na Model, View i Controller. U konkretnom slučaju zbog dva stanja aplikacije koristio sam dva controllera nad jednim view-om. View je prezentacijski dio i prikazuje korisniku informacije dok controlleri obrađuju korisnikov unos i komuniciraju sa modelom.

