1. **Singleton** :

ParametriSG korišten jer je bio potreban samo 1 instanca te klase, preuzima argumente iz naredbenog retka te dalje distribuira potrebitim klasama njihove informacije (ConcUredjajima nazive datoteka, Generatoru brojeva sjeme, ili signal za generiranje istog I sl)

GenBrojevaSG prema uputstvu iz zadaće. Potrebna točno 1 instanca I globalno dostupna.

ListaSvegaSG potrebna kako bi u svakom trenutku bile dostupne sve liste mjesta, senzora, aktuatora I njihovih pripadajućih metoda.

IspisUpisSG korištena radi globalnog upisa I ispisa podataka, operacija ispisivanja datoteke I uloge Viewa. F\_Simplifierradi praktičnosti korištenja Façade uzorka. Originator I Pazikuca (CareTaker) kako bi zadržali svoju jedinu formu I podatke, te bili globalno dostupni za korištenje potpune Memento realizacije.

1. **Factory Method**

Premda su korišteni I oblici buildera, oni su tu da pomognu klasi ConcUredjajiFM da uz naziv datoteke I tip prekidača koji ide uz naziv, pomoću sučelja definiranog u klasi UredjajiFM jednim pozivom stvara 3 objekta po potrebi (npr. stvoriObjekt(“DZ\_3\_mjesta”,-m), isto za senzore ili aktuatore.

* 1. Koristi se I oblik buildera za egzekuciju ove zamisli.

1. **Prototype**

Koristi se za stvaranje novog starog objekta koji po nekom kriteriju ne valja. Prilikom zamjene senzora ili aktuatora, koristim metodu kloniraj() iz sučelja ProtoPlugInIF koja vraća object, a po potrebi se onda transformira u senzor ili aktuator.

1. **Iterator** je zasebna klasa sa definiranim metodama iz nadklase IteratorIF. Služi za kretanje algoritma
2. **Façade** za pojednostavljenje podsustava, odnosno praktičniji doziv “raštrkanih” funkcija širom dijagrama, praktički ima I ulogu medijatora budući View ne bi smio komunicirati direktno sa Modelom
3. **Observer** za dojavu aktuatoru promjene stanja senzora. Prilikom svake promjene vrijednosti senzora, actuator dobiva “notification” od senzora da promjeni stanje na račun promjene senzorove vrijednosti.
4. **MVC** uzorak po uputama korišten za strukturiranje arhitekture. Zamišljen kao restrukturiranje postojećeg sustava u 3 pool-a uloga, gdje je view zadužen za ispis na ekran (IspisUpisSG), controller pool sastoji se od nekoliko klasa gdje su glavne F\_Simplifier I C\_Parametri koji imaju ulogu medijatora, odnosno obrade parametara koje korisnik unese I potom pozivanja potrebnih funkcija. F\_Simplifier samo “urednije” centralizira sve pozive.
5. **Command** je korišten u tandemu sa Mementom gdje su naredbe Create i RestoreMemento realizirane kao objekti te je sve što je potrebno za realizaciju tih akcija, samo instancirati objekt i specifičnu akciju .execute() pokrenuti.
6. **Memento** je korišten u svrhu pohrane prijašnjih stanja liste mjesta te za vraćanje istih. Najpraktičniji uzorak za takvu vrstu problema.
7. **ChainOfResponsibility** ostao kao način različitog računanja statistika, klasa Stat1 koristi jedne metode, a Stat2 druge.Obje izbacuju druge analize.

**Vlastita funkcionalnost**

* Chain of responsibility uzorak implementiran je uz 2 klase (statistika, statistika2) tako da računa različite statističke podatke na različite načine i pokazuje različite rezultate. Ne računaju obje klase istu stvar.

**Razlika u odnosu na prijašnju zadaću**

Dodan je Memento, upotpunjen je Command uzorkom te su dodane funkcionalnosti za zamjenu uređaja, funkcionalnosti TS, TA i SM sa odgovarajućim argumentima. Izmjenjena je struktura klase Mjesta radi potreba stvaranja stablaste strukture. Poboljšan je prolaz algoritma/ciklusa. Odrađena je mehanika ograničenja i manipulacija kolekcijama za različite modele.

