Ayana cvjećara

Kreacijski patterni

*boldirane pattern-e ćemo sigurno primijeniti

1. Singleton pattern

Singleton pattern koristimo u slučaju kada nam je potrebno da se klasa/objekat može instancirati samo jednom I kada nam je potrebna jedinstvena kontrola pristupa.

U našem sistemu, mogle bismo primijeniti navedeni pattern ukoliko bismo imale nekoliko zaposlenika među kojima bi jedna bio administrator/admin koji bi imao određene privilegije u odnosu na ostale zaposlenike. Naprimjer, admin može pogledati I preuzeti izvještaj, dok ostale zaposlenike poput dostavljača I sl. Izvještaj ne zanima. Trenutno, radi jednostavnosti, mi imamo jednog zaposlenika koji naravno ima pristup izvještaju, ali u slučaju potrebe proširenja sistema, mogle bismo na ovaj način iskoristiti Singleton pattern.

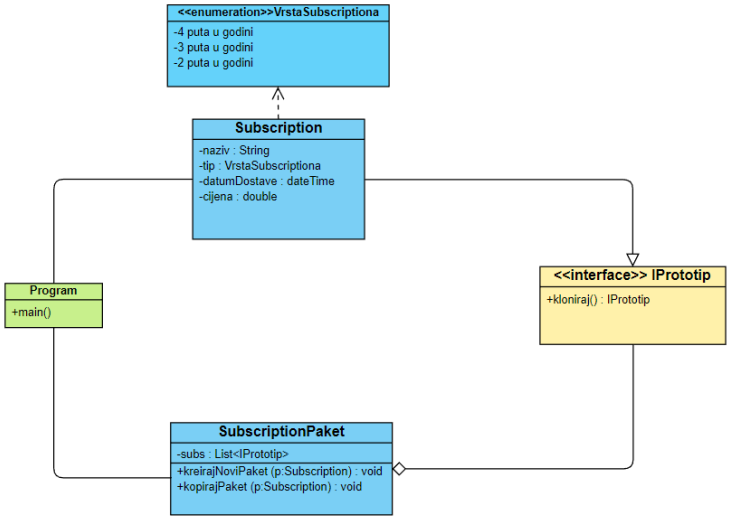
Ipak smo implementirale Singleton pattern za korištenje eksternog servisa *Mailgun da se ne bi kreiralo više istih instanci jer onda dolazi do konflikta.*

2. Prototype pattern

Navedeni pattern koristimo ako nam je potrebno kopiranje odnosno kloniranje, površinsko ili duboko.

Ovaj pattern može naći primjenu u našem sistemu nad apstraktnom klasom Osoba koja ima izvedene klase korisnik I zaposlenik. Ukoliko se prilikom registracije desi da korisnik unese podatke koji dijelom odgovaraju podacima nekog već registrovanog korisnika (isto ime I prezime naprimjer), umjesto kreiranja novog objekta, možemo klonirati postojeću instancu I izmijeniti odgovarajuće podatke koji se razlikuju.

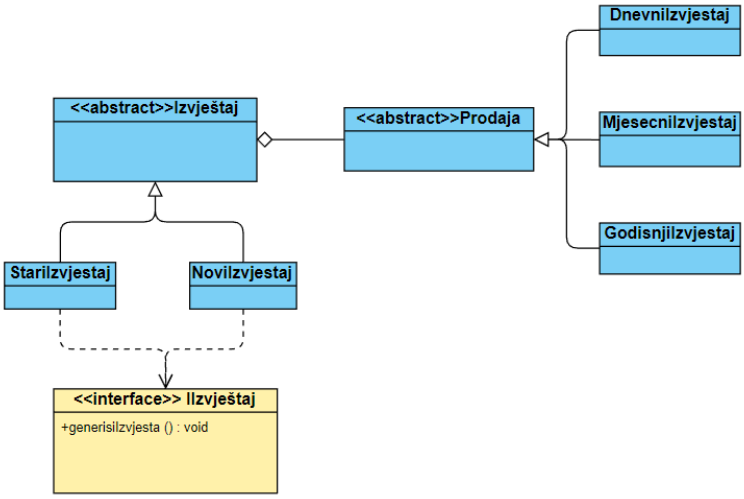
Druga, bolja primjena ovog patterna je vezana za ponudu subscription paketa. Postoje 3 vrste paketa koje su unaprijed definisane tako da su to jedine opcije bez razlike prilikom odabira od različitih korisnika pa mi možemo koristiti prethodnu instancu uz moguću izmjenu sadržaja opcionalnog personal card-a.



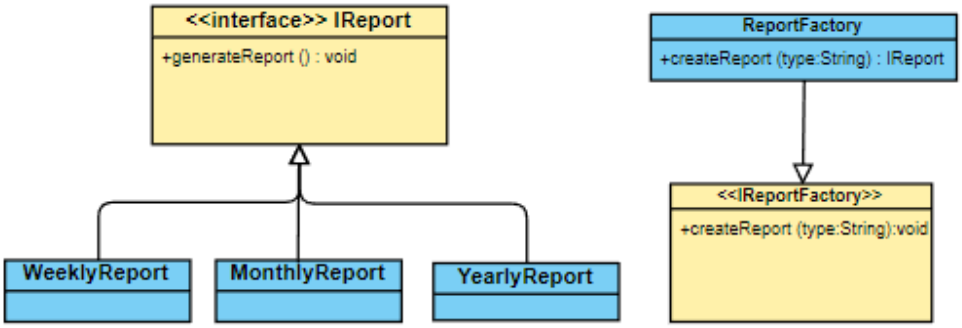
3. *Factory method pattern*

Factory method pattern koristimo za kreiranu instancu od više izvedenih klasa. Realizujemo ga pomoću iProduct interface-a I klasa ProductA, ProductB... koje implementiraju interface.

Primjenu u našem sistemu možemo ostvariti preko funkcionalnosti izvještaja. Postoje tri vrste izvještaja koje sistem generiše zavisno od zahtjeva zaposlenika: dnevni, mjesečni I godišnji. Osnova može biti klasa apstraktna klasa Izvještaj koja može instancirati tri različita izvještaja.



Nakon implementacije, dijagram izgleda ovako:



4. *Abstract factory pattern*

Ovaj pattern se koristi za kreiranje familije povezanih objekata bez konkretnih klasa I odvajanje definicije klase/produkta od korisnika.

Ideja za primjenu ovog pattern-a na naš sistem bi mogla biti klasa Popust. Tu klasu bi naslijedile klase koje bi se razlikovale ovisno da li se radi o popustu koji se ostvaruje preko koda iz email-a ili popustu koji se odnosi na posebne prilike I važne datume.

5. *Builder pattern*

Builder pattern se koristi za odvajanje specijalnih kompleksnih objekata od stvarne konstrukcije. Možemo koristiti klasu Subscription (analogno klasi Director na predavanju), interface IBuilder te klase ConcreteBuilder I Product.

U našem sistemu, ovaj pattern bismo isto mogle primijeniti na funkcionalnost subscription poput Prototype patern-a. Kao što sam ranije spomenula, ponuda subscriptions paketa je fiksna. Dodatna pogodnost u ovom slučaju je personal card koji ne mora biti odabran I u tom slučaju bi se narudžba automatski generisala. Ukoliko kupac odabere da želi personal card, on piše poruku I kartica se dodaje uz narudžbu, naravno bez promjene cijene.