Paterni ponašanja

U osnovne paterne ponašanja spadaju: Strategy patern, State patern, TemplateMethod patern, Observer patern, Iterator patern. U nastavku ćemo objasniti svaki patern pojedinačno.

Strategy patern

Strategy patern izdvaja algoritam iz matične klase i uključuje ga u posebne klase. Pogodan je kada postoje različiti primjenjivi algoritmi za neki problem. Kada bi se različite metode klase „Korisnik“ drugačije odvijale za sve podklase, tada bi se ovaj patern mogao iskoristiti. Kako to nije slučaj kod nas, i kako nema drugih nasljeđivanja primjena ovog paterna bi bila suvišna.

State patern

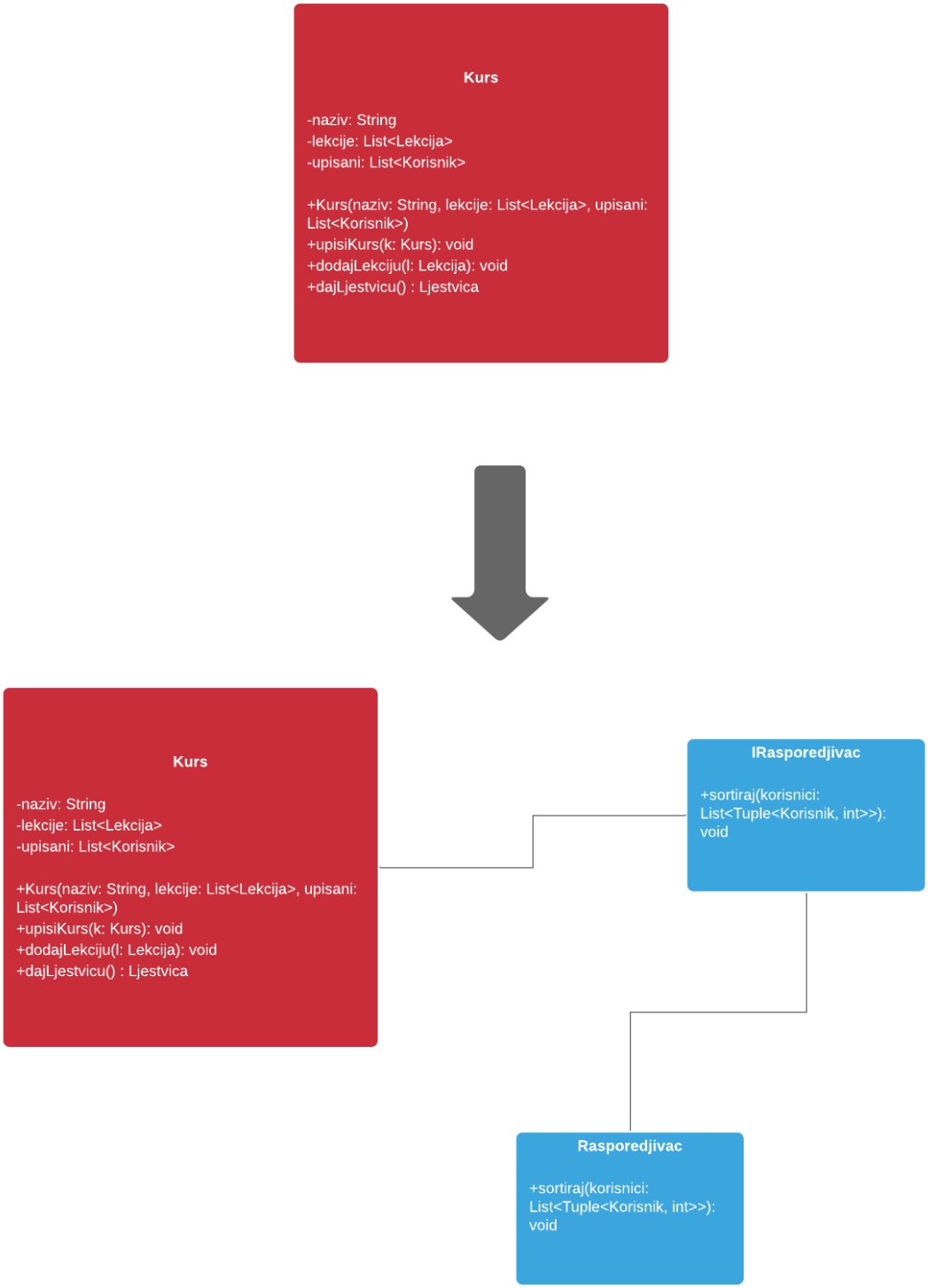
State patern je dinamička verzija Strategy paterna, što bi značilo da mijenja način ponašanja na osnovu trenutnog stanja. U našem projektnom zadatku se ovaj patern ne primjenjuje s obzirom da nemamo situaciju da objekat mijenja način ponašanja, tj. nema promjene stanja. Kada se posjetilac uloguje, on ne mijenja stanje u neko drugo. Također, isto vrijedi i prilikom registracije i prijave.

Observer patern

Observer patern uspostavlja relaciju između objekata. Kada jedan objekat promijeni stanje, drugi zainteresirani objekti se obavještavaju. Ovaj patern smo mogli primijeniti na naš projektni zadatak, ali zbog potreba tutorijala nismo. Slučaj u kojem bi se mogao primijeniti je ukoliko bi se korisnikovo stanje na ljestvici promijenilo, on bi bio obaviješten.

TemplateMethod patern

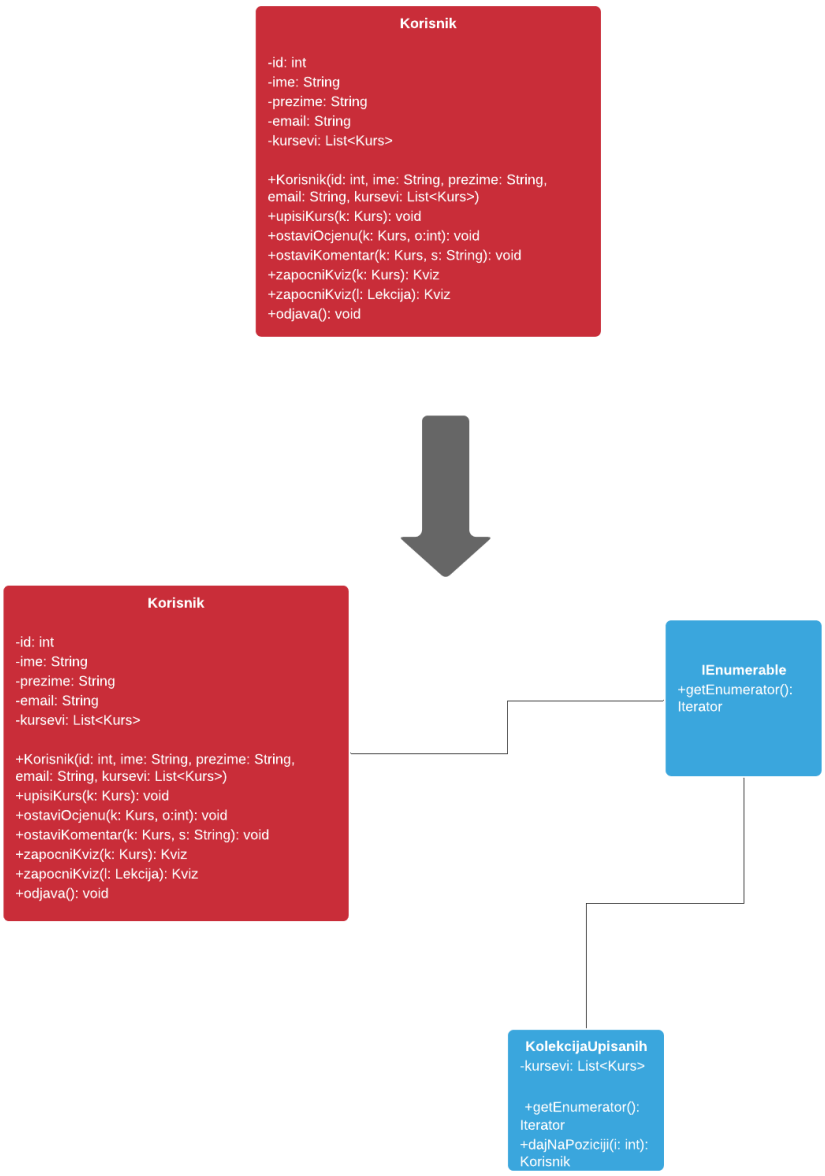
vaj patern omogućava izdvajanje određenih koraka algoritama u odvojene podklase. U našem primjeru konkretno, mi smo primijenili ovaj patern koristeći metodu „**dajLjestvicu(): Ljestvica**“ kao TemplateMethod. Operacija koja se primjenjuje u Iraporedjivac-u i Rasporedjivac-u je „**sortiraj(korisnici: List<Tuple<Korisnik, int>>): void“**, gdje argument funkcije koja sortira predstavlja listu uređenih parova koji predstavljaju korisnika zajedno sa njegovim poenima. Na slici ispod vidimo stanje prije i poslije upotrebe ovog paterna:



Slika 1: primjena TemplateMethod paterna na projektni zadatak

Iterator patern

Uloga ovog paterna je da omogući sekvencijalni pristup elementima kolekcije bez poznavanja kako je kolekcija strukturirana. Interfejs koji smo dodali je IEnumerable koji sadrži metodu „**getEnumerator(): Iterator**“, a kolekcija „KolekcijaUpisanih“ nasljeđuje tu klasu. Naime, primarni cilj primjene ovog paterna jeste da imamo iterativno idemo kroz upisane kurseve jednog korisnika. Na slici smo prikazali primjenu ovog paterna:



Slika 2: prikaz primjene Iterator paterna