Univerzitet u Sarajevu

Elektrotehnički fakultet u Sarajevu

Paterni ponašanja (komunikacije)

FitnessFusion

Predmet: Objektno orijentisana analiza i dizajn

Naziv tima: Time – out

Članovi tima: Ahmetović Zerina

Bajrović Taner

Bosno Hamza

Brčaninović Hasan

Peljto Emina

Sarajevo, jun 2023.

Paterni ponašanja (komunikacije):

Paterni ponašanja (komunikacije) su namjenjeni da uspostave komunikaciju između objekata, rasporede odgovornosti i organizuju algoritme koji se koriste u sistemu. Ovom skupinom paterna se postiže bolja povezanost između objekata kao i željeni način funkcionisanja sistema na organizovan način koji poštuje odlike dobrog dizajna.

U sklopu ovog dokumenta će se predstaviti dva paterna ponašanja koja će se iskoristi u ovom projektu, te će se razmotriti i potencijalna primjena drugih paterna ponašanja.

Paterni koji će se upotrijebiti u ovom projektu i detaljnije opisati su *Mediator* i *Observer*.

Neki od problema koji su motiv za upotrebu paterna ponašanja su:

- Uspostavljanje komunikacije između trenera i korisnika prilikom razmatranja rasporeda.
- Uspostavljanje komunikacije putem notifikacija koje obavještavaju korisnika o izmjenama.
- Omogućavanje različitih načina plaćanja.
- Upotreba više metoda za obradu niza različitih vrijednosti na različite načine.

U nastavku slijedi detaljniji opis prethodno navedenih paterna koji su se iskoristili, kao i potencijalna mjesta na kojima bi se mogli upotrijebiti i drugi paterni u našem sistemu.

Mediator dizajn pattern:

Mediator dizajn patern omogućava reduciranu i organizovanu komunikaciju između zavisnih objekata putem posrednog objekta ili mediatora.

Ovaj patern se koristi u slučaju kada trener vrši prijedlog izmjene korisnikovog rasporeda, a korisnik može ili da prihvati ili da odbije taj prijedlog. Trener i korisnik mogu komunicirati kroz medijator koji će olakšati proces izmjene rasporeda i uspostaviti koordinaciju u komunikaciji i donošenju odluka.

Primjer:

Ovaj primjer predstavlja princip koji se jednim dijelom koristi u našem sistemu za prethodno navedeni dio u sistemu.

```
interface ScheduleMediator {

    fun recommendScheduleChange(trainer: Trainer,
newSchedule: Schedule)

    fun acceptScheduleChange(client: Client)

    fun refuseScheduleChange(client: Client)

}
```

Observer dizajn pattern:

Ovim paternom se uspostavljaju relacije između objekata na način da jedan objekat promjeni stanje drugih objekata koji trebaju tu promjenu.

Ovaj patern možemo iskoristiti u scenariju kada korisnik treba dobiti notifikaciju da je trener predložio izmjenu rasporeda.

Primjer:

Ovaj primjer predstavlja princip koji se jednim dijelom koristi u našem sistemu za prethodno navedeni dio u sistemu.

```
interface ScheduleObserver {
    fun onScheduleChangeRecommended(newSchedule: Schedule)
}

interface ScheduleObservable {
    fun addObserver(observer: ScheduleObserver)
    fun removeObserver(observer: ScheduleObserver)
    fun notifyScheduleChangeRecommended(newSchedule:
Schedule)
}
```

U ovoj implementaciji trener predlaže izmjenu rasporeda tako što pozove recommendScheduleChange.

Korisnik može biti subscribed na ScheduleObservable tako što će implementirati ScheduleObserver interfejs i registrovati sebe koristeći addObserver.

Strategy dizajn patern:

Strategy patern omogućava upotrebu više različitih algoritama za neku operaciju u sistemu. Potencijalna primjena ovog algoritma je u dijelu sistema koji se odnosi na način plaćanja. Prilikom upotrebe sistema korisnik vrši uplatu članarine na jedan od nekoliko ponuđenih načina. To omogućava strategy patern koji neovisno od načina obavlja isti zadatak.

State dizajn patern:

Ovim paternom se omogućava dinamična promjena ponašanja objekta tokom vremena. Potencijalna primjena ovog paterna bi bila moguća kad bi sistem vršio različite proračune rezultata u zavisnosti od toga u kojim je stanjima korisnik. Obzirom da korisnik tokom praćenja određenog programa aktivnosti prelazi u različita stanja tokom vremena, tada bi sistem trebao da omogući različite načine da se rezultati računaju, što bi davalo preciznije rezultate.

Template dizajn patern:

Template dizajn patern omogućava da se uspostavi jedinstven template za izvršavanje nekog zadatka u sistemu koji se može primjeniti u zavisnosti od klasa koje definišu njegovo implementiranje. Ovaj dizajn paterni svoju potencijalnu primjenu pronalazi u dijelu sistema koji se odnosi na način plaćanje. Obzirom da su Strategy i Template paterni slični, u zavisnosi od konkretene implementacije je potrebno detaljnije analizirati koji od njih bi predstavljajo bolje rješenje.

Iterator dizajn patern:

Ovim paternom se postiže iterativni pristup elementima kolekcije objekata pri čemu je sama struktura kolekcije zanemarana. Ovaj patern se indirektno korisiti prilikom upotrebe bibliotečnih tipova podataka kao što su liste. Također, potencijalna primjena ovog paterna je moguća u slučaju da se klasa parametara organizuje kao struktura stablo, pri čemu bi se iterator koristio za pristup elementima i obilazak elemanata u tako struktuiranoj kolekciji.

Command dizajn patern:

Command dizajn patern omogućava zadavanje zahtjeva za izvršavanjem zadatka na enkapsuliran način za odgođeno izvršavanje. Ovaj dizajn paterni svoju potencijalnu primjenu u sistemu može da pronađe ako se koristi za funkcionalnosti poput naruđbe. U slučaju da se ovaj sistem proširi sa funkcionalnostima online shop-a, tada bi se ovaj patern mogao da primjeni.