Sveučilište u Zagrebu
Fakultet organizacije i informatike
Pavlinska 2, 42000 Varaždin

Obrazac za zadaću na predmetu "**Uzorci dizajna**"

Ime i prezime studenta/ice: Ivan Šiser

Matični broj: 0016123395

Dio A. Osnovni podaci o zadaći

R.br.	Pitanje	Odgovor	
1.	Grupa na seminaru:	G3	
2.	Broj i naziv zadaće:	3	Iznajmljivanje električnih vozila.
3.	Procjena vremena za realizaciju bez decimala):	10 sati	
4.	Procjena % završenosti (bez decimala):	25 / 100%	
5.	Procjena bodova za izradu zadaće (1 decimala):	4 / (DZ1 - 12, DZ2 – 13, DZ3 – 15)	
6.	Žalim prezentirati zadaću:	Ne	
7.	Koji dijelovi iz opisa zadaće nisu realizirani:	- Sve što je navedeno da nije realizirano u zadaći 2 - Nije implementiran uzorak Iterator, kao ni poseban Chain of Responsibility za vođenje evidencije oko računa - Nije implementirana MVC arhitektura - Ne postoji opcija spremanja output-a u datoteku	
8.	Postoji li dio zadaće koji vrijedi posebno istaknuti i zašto:	Htio bih istaknuti da je aplikacija testirana pomoću datoteka s podacima postavljenim na moodlu sustavu i da sve uredno radi. U slučaju testiranja s nekim drugim podacima moguće su pogreške koje nisu regulirane.	
9.	Postoje li dijelovi zadaće koji imaju pogrešku u radu i koje:	Pogreška ima prilikom ispisivanja stanja u aktivnosti 6., postavljeno je da ispisuje stanje broja mjesta, broja raspoloživih i neispravnih vozila prema lokacijama, a ne prema tipu vozila.	
10.	Da li ste koristili posebne biblioteke u realizaciji zadaće izvan spomenutih na nastavi:	Ne	
11.	Da li ste koristili programska rješenja ili dijelove programskog koda od drugih kolega:	Ne	

Dio B. Dokumentacija rješenja zadaće

Naziv uzorka dizajna	Klase koje sudjeluju u uzorku dizajna	Opis razloga odabira uzorka dizajna
Singleton - 2x	**Paket – stvaranjeAktivnosti** BazaPodataka.cs VirutalnoVrijeme.cs	Baza podataka kao stanje programa potrebno je instancirati samo jedanput što ovaj uzorak upravo i osigurava. Singleton omogućuje također globalni pristup do varijabli odnosno listi podataka iz klase što također pogoduje ovoj implementaciji, pošto podatke koristimo u skoro pa svakoj klasi.
Builder	**Paket – stvaranjeAktivnosti** AktivnostBuildDirector.cs AktivnostBuilder.cs AktivnostiBuilderImpl.cs	Kako u samoj aplikaciji razlikujemo više aktivnosti odnosno komandi s različitim brojem argumenata, builder je odlična solucija za kreiranje objekta s puno opcionalnih argumenata, također takvim kreiranjem postaje neovisan o svojim dijelovima.
Factory Method – 2x	**Paket – ucitavanjeDatoteka** (ICitanje.cs, CitanjeCreator.cs, CitanjeVozila..) **Paket – obradaKomandi** (IObrada.cs, NajamVozila.cs..)	U aplikaciji sam prepoznao dvije funkcionalnosti koje moraju biti implementirane s raznim varijacijama – učitavanje datoteka, kao i obrada komandi pa je iz toga razloga factory method implementiran dva puta. Podklase u ovom slučaju određuju na temelju parametara koju klasu točno treba instancirat.
Prototype	Vozilo.cs	Prilikom učitavanja kapaciteta, potrebno ih je napuniti s brojem raspoloživih vozila odgovarajućeg tipa. Iz toga razloga sam ovdje prepoznao uzorak dizajna prototype, pomoću kojega jedno stvoreno vozilo mogu klonirati i na taj način napuniti liste u kapacitetima. Prilikom stvaranja pojedinog klona, automatski sam dodjeljivao jednoznačni identifikator.
Template Method	**Paket – pokretanjePrograma** (NacinIzvrsavanja.cs, PutemKonfiguracije.cs, PutemOpcija.cs)	Prilikom inicijalizacije programa potrebno je definirati dva načina na koji se program može izvršiti: putem opcija i naziva datoteka ili putem konfiguracijski datoteka. Putem template methoda realizirano je na koji će se način pokrenuti program s obzirom na unesene argumente (na temelju analize unesenih argumenata pokreće se određena klasa koja određuje rad programa).
Observer	**Paket – izdavanjeRacuna** (Publisher.cs, Subscriber.cs) **Paket – modeli** (Lokacija.cs)	Kako u zadatku stoji da račun ne smije biti direktno putem objekta povezan s lokacijom ovdje je korišten uzorak dizajna Observer. Svaki puta kada se izdaje račun, svi pretplatnici koji promatraju promjene u listi računa dobivaju obavijest i putem toga ažuriraju svoje zarade.
Composite	**Paket – strukturaOrganizacije** (OrganizationComponent.cs, OrganizationLocation.cs, OrganizationUnit.cs)	Kako u zadatku postoji organizacijska struktura koja se temelji na organizacijskim jedinicama, lokacijama i ishodišnoj tvrtki i u pojedinim aktivnostima potrebno ju je ispisat, za ovaj problem idealno je rješenje Composite. Pomoću njega izgrađeno je stablo strukture.
Chain of Responsibility	**Paket – ispisPodatakaOrganizacije** (IHandler.cs, Stanje.cs, Struktura.cs, Zarada.cs)	Kako sam prethodno koristio Composite za izgradnju, preporuka je da se za njega koristi Chain of Responsibility kako bi svaka organizacijska jedinica svoj zahtjev proslijedila u lanac koji procesuira zahtjev.
State	**Paket – stanjeVozila** (VoziloIznajmljeno.cs, VoziloSlobodno.cs, VoziloState.cs..) **Paket – modeli** (Vozilo.cs)	State omogućuje upravljanje sa stanjima, u ovom slučaju upravljanjem stanjima vozila (pojedine aktivnosti s vozilom mogu biti pozvane samo ako odgovara stanje vozila čime se eliminiraju provjere na samom vozilu).

Dio C. Opis promjena u odnosu na prethodnu zadaću (osim kod 1. zadaće)

1) Dodan je uzorak State - upravljanje stanjima vozila
    a. Sve do sada implementirane opcije rada s vozilima prebačene su iz uzorka dizajna Builder tako da odgovaraju radu unutar određenih stanja vozila

2) Realizirane su nove aktivnosti (9., 10. i 11.)
3) Realizirane su dodatne provjere ispravnosti podataka u datotekama uz ispisivanje retka s pogreškom

Dio D. Dijagram klasa s naglašavanjem klasa koje sudjeluju u pojedinom uzorku dizajna