Sveučilište u Zagrebu

Fakultet organizacije i informatike

Pavlinska 2, 42000 Varaždin

Obrazac za zadaću na predmetu "**Uzorci dizajna**" ak. god. 2024./2025.

Ime i prezime studenta/ice: Ivan Milanović-Litre

Matični broj: 0016150879

**Dio A. Osnovni podaci o zadaći**

|  |  |  |  |
| --- | --- | --- | --- |
| **R.br.** | **Pitanje** | **Odgovor** | |
| 1. | Grupa na seminaru: | G1 | |
| 2. | Broj i naziv zadaće: | 3 | Željeznički promet s voznim redom |
| 3. | Procjena vremena za realizaciju bez decimala): | 18 sati | |
| 4. | Procjena % završenosti (bez decimala): | 85 / 100% | |
| 5. | Procjena bodova za izradu zadaće ( 1 decimala): | 11 / (DZ3 - 15) | |
| 6. | Žalim prezentirati zadaću: | NEMA PREZENTACIJE ZADAĆE! | |
| 7. | Koji dijelovi iz opisa zadaće nisu realizirani: | -Iz prethodne zadaće nije realizirana komanda IVI2S  -Nema vlastite funkcionalosti Command | |
| 8. | Postoji li dio zadaće koji vrijedi posebno istaknuti i zašto: | Ne | |
| 9. | Postoje li dijelovi zadaće koji imaju pogrešku u radu i koje: | Pri pokretanju zadaće javlja da su određeni redci prazni, što je rezultat pogrešaka pri učitavanju csv. | |
| 10. | Da li ste koristili tuđi programski kod u realizaciji zadaće izvan spomenutih izvora na nastavi: | -Korištene su generičke klase za Factory Method. Naime,  postoji CitacPodataka kojeg nasljeđuju CitacVlakova,  CitacPruga... Citaci su zamišljeni da učitavaju podatke te  sukladno podatcima koji su učitali generiraju određene objekte  kao što su lista pruga i lista vlakova. Riječ je o konceptu kojeg  smo učili u sklopu predmeta Strukture podataka i algoritmi.  Dodatni primjeri u Javi su proučeni na slijedećoj stranici :  <https://www.geeksforgeeks.org/generics-in-java/>.  -Za otkrivanje dana u tjednu koristio se slijedeći izvor: <https://www.geeksforgeeks.org/dayofweek-getvalue-method-in-java-with-examples/> | |
| 11. | Da li ste koristili programska rješenja ili dijelove programskog koda od drugih kolega: | Ne | |

**Dio B.1. Dokumentacija rješenja 2. zadaće (kopirano i nepromijenjeno)**

|  |  |  |  |
| --- | --- | --- | --- |
| **Naziv uzorka dizajna** | **Klase koje sudjeluju u uzorku dizajna i u kojim ulogama** | **Sta tus[[1]](#footnote-1)** | **Opis razloga odabira uzorka dizajna** |
| Singleton | ZeljeznickaTvrtka | S | Tvrtka mora biti jedinstvena. Prilikom  dodavanja elemenata u tvrtku (pruga, stanice,  vozila) mora se osigurati jedinstvne  infrastrktura. |
| Builder | ZeljeznickaStanica – u  sebi sadrži  ZeljeznickaStanicaBuilder,  PrijevoznoSredstvo – u  sebi sadrži  PrijevoznoSredstvoBuilder  , Pruga – u sebi sadrži  PrugaBuilder | S | Korištenjem Buildera omogućuje se fleksibilno i  jasno kreiranje objekata, gdje se samo ključni  atributi postavljaju obavezno, dok su ostali  atributi opcionalni. Ovaj uzorak također  povećava čitljivost i sigurnost koda, budući da  se objekti postavljaju kao nepromjenjivi nakon  kreiranja. |
| FactoryMethod | CitacPodataka –  apstraktna klasa koju  nasljeđuju:  CitacKompozicija,  CitacPrijevoznihSredstava  , CitacPruga,  CitacZeljeznickihStanica,  CitacVlakova  ----------------------------  CitacPodatakaFactory –  apstraktna klasa koju  nasljeđuju:  CitacKompozicijaFactory,  CitacPrijevoznihSredstava  Factory,  CitacPrugaFactory,  CitacZeljeznickihStanicaF  actory,  CitacVlakovaFactory | P | Factory Method je ovdje korišten jer postoji  potreba za stvaranjem različitih vrsta objekata  za učitavanje podataka (npr., CitacKompozicija,  CitacPrijevoznihSredstava, CitacPruga,  CitacZeljeznickihStanica). Svaki od ovih čitača  ima specifičnu logiku za obradu i parsiranje  podataka iz izvora podataka, a Factory Method  omogućava da kreiranje tih objekata bude  centralizirano i organizirano. Korištenjem  apstraktne klase CitacPodatakaFactory koja  definira metodu za kreiranje objekata čitača,  omogućujemo jednostavno proširenje sustava  novim tipovima čitača bez izmjena postojećeg  koda. Svaki konkretni CitacFactory (kao što su  CitacKompozicijaFactory,  CitacPrijevoznihSredstavaFactory i ostali) može  kreirati odgovarajući čitač, osiguravajući  jasnoću i održivost koda |
| Composite | KomponentaVoznogReda  --------------------------  VozniRed  Vlak  Etapa | N | VozniRed u sebi sadrži više objekata Vlak, a  svaki Vlak sadrži jednan ili više objekata Etapa.  Pošto je ovaj odnos uređen kao stablo korišten  je Composite. Za sučelje je korištena  KomponentaVoznogReda koju nasljeđuju  VozniRed, Vlak i Etapa. Popis svih vlakova se  učitava iz csv-a uz pomoč čitača vlakova tako  da se za svaki Vlak kreiraju Etape a svakoj Etapi  dodijele pripadajuće stanice. Zatim se u  ZeljeznickaTvrtka VozniRed puni svim  vlakovima. |
| Observer | Subject  -----------  Korisnik  Observer  -----------  Vlak  ZeljeznickaStanica | N | Pošto je potrebno pokrenuti simulaciju za  određeni vlak te o prolasku kroz sve stanice ili  pojedinu stanicu implementiran je Observer.  Unutar apstraktnih klasa definirane su metode  koje omogućuju da Observer bude  implementiran te Vlak i ZeljeznickaStanica  nasljeđuju Subject a Korisnik Observer.  Unutra vlaka postoji metoda koja pokreće  simulaciju te o određenoj radnji informira  pojedinog korisnika. |

**Dio B.2. Dokumentacija rješenja 3. zadaće**

|  |  |  |  |
| --- | --- | --- | --- |
| **Naziv uzorka dizajna** | **Klase koje sudjeluju u uzorku dizajna i u kojim ulogama** | **Sta tus[[2]](#footnote-2)** | **Opis razloga odabira uzorka dizajna** |
| Singleton | ZeljeznickaTvrtka | S | Tvrtka mora biti jedinstvena. Prilikom  dodavanja elemenata u tvrtku (pruga, stanice,  vozila) mora se osigurati jedinstvne  infrastrktura. |
| Builder | ZeljeznickaStanica – u  sebi sadrži  ZeljeznickaStanicaBuilder,  PrijevoznoSredstvo – u  sebi sadrži  PrijevoznoSredstvoBuilder  , Pruga – u sebi sadrži  PrugaBuilder | S | Korištenjem Buildera omogućuje se fleksibilno i  jasno kreiranje objekata, gdje se samo ključni  atributi postavljaju obavezno, dok su ostali  atributi opcionalni. Ovaj uzorak također  povećava čitljivost i sigurnost koda, budući da  se objekti postavljaju kao nepromjenjivi nakon  kreiranja. |
| FactoryMethod | CitacPodataka –  apstraktna klasa koju  nasljeđuju:  CitacKompozicija,  CitacPrijevoznihSredstava  , CitacPruga,  CitacZeljeznickihStanica,  CitacVlakova  ----------------------------  CitacPodatakaFactory –  apstraktna klasa koju  nasljeđuju:  CitacKompozicijaFactory,  CitacPrijevoznihSredstava  Factory,  CitacPrugaFactory,  CitacZeljeznickihStanicaF  actory,  CitacVlakovaFactory | S | Factory Method je ovdje korišten jer postoji  potreba za stvaranjem različitih vrsta objekata  za učitavanje podataka (npr., CitacKompozicija,  CitacPrijevoznihSredstava, CitacPruga,  CitacZeljeznickihStanica). Svaki od ovih čitača  ima specifičnu logiku za obradu i parsiranje  podataka iz izvora podataka, a Factory Method  omogućava da kreiranje tih objekata bude  centralizirano i organizirano. Korištenjem  apstraktne klase CitacPodatakaFactory koja  definira metodu za kreiranje objekata čitača,  omogućujemo jednostavno proširenje sustava  novim tipovima čitača bez izmjena postojećeg  koda. Svaki konkretni CitacFactory (kao što su  CitacKompozicijaFactory,  CitacPrijevoznihSredstavaFactory i ostali) može  kreirati odgovarajući čitač, osiguravajući  jasnoću i održivost koda |
| Composite | KomponentaVoznogReda  --------------------------  VozniRed  Vlak  Etapa | S | VozniRed u sebi sadrži više objekata Vlak, a  svaki Vlak sadrži jednan ili više objekata Etapa.  Pošto je ovaj odnos uređen kao stablo korišten  je Composite. Za sučelje je korištena  KomponentaVoznogReda koju nasljeđuju  VozniRed, Vlak i Etapa. Popis svih vlakova se  učitava iz csv-a uz pomoč čitača vlakova tako  da se za svaki Vlak kreiraju Etape a svakoj Etapi  dodijele pripadajuće stanice. Zatim se u  ZeljeznickaTvrtka VozniRed puni svim  vlakovima. |
| Observer | Subject  -----------  Korisnik  Observer  -----------  Vlak  ZeljeznickaStanica | S | Pošto je potrebno pokrenuti simulaciju za  određeni vlak te o prolasku kroz sve stanice ili  pojedinu stanicu implementiran je Observer.  Unutar apstraktnih klasa definirane su metode  koje omogućuju da Observer bude  implementiran te Vlak i ZeljeznickaStanica  nasljeđuju Subject a Korisnik Observer.  Unutra vlaka postoji metoda koja pokreće  simulaciju te o određenoj radnji informira  pojedinog korisnika. |
| State | RelacijaStatus  -----------------  Ispravna  Kvar  Testiranje  Zatvorena  ---------------  Relacija | N | State uzorak dizajna omogućio je jednostavno i učinkovito upravljanje statusima relacija unutar pruge. Svaka relacija ima svoj status, poput Ispravna, Kvar, Testiranje ili Zatvorena, a svaki status je implementiran kao zasebna klasa koja definira svoje ponašanje. Na primjer, pravila prijelaza između statusa jasno su definirana, što osigurava kontrolu nad promjenama i sprječava nevažeće prijelaze. State omogućuje i jednostavnu provjeru dozvoljenosti putovanja relacijom, čime se pojednostavljuje logika upravljanja vlakovima. Ovaj pristup izbjegava složene uvjete i centralizira logiku u odgovarajuće klase. Također, proširivost je jednostavna – dodavanje novog statusa ne utječe na postojeće klase. State je omogućio modularnost, održivost i fleksibilnost, olakšavajući rad s kompleksnim sustavom pruga i relacija. |
| Memento | Karta  ---------------  KartaMemento  UpraviteljKartama | N | Memento uzorak dizajna korišten je za spremanje i dohvaćanje podataka o kupljenim kartama u sustavu, omogućujući čuvanje trenutnog stanja bez direktnog izlaganja unutarnjih podataka objekta Karta. Svaka karta se prilikom kupnje sprema kao KartaMemento, koja pohranjuje ključne informacije o karti poput oznake vlaka, relacije, datuma, vremena, cijene i načina plaćanja. UpraviteljKartama služi kao centralni upravitelj za rad s ovim memento objektima – omogućuje pohranjivanje novih karata, dohvaćanje povijesnih podataka ili ispis posljednjih kupljenih karata. Ovaj obrazac omogućio je razdvajanje logike pohrane od same logike upravljanja kartama, osiguravajući sigurno upravljanje povijesnim podacima. Implementacija je također olakšala pregled i analizu kupljenih karata kroz različite funkcionalnosti sustava, bez narušavanja kapsulacije glavne klase Karta. |
| Strategy | PlacanjeStrategy  --------------------  PlacanjeBlagajnaStrategy  VlakPlacanjeStrategy  WebMobPlacanjeStrategy --------------------------Karta | N | Strategy uzorak dizjna omogućio je fleksibilno i dinamično upravljanje različitim načinima plaćanja za karte. Umjesto dugih i nepreglednih if-else uvjeta, svaki način plaćanja, poput Plaćanje na blagajni, Plaćanje u vlaku ili Plaćanje putem web/mobilne aplikacije, implementiran je kao zasebna strategija u odgovarajućim klasama. Svaka karta ima svoj definirani Strategy koji određuje kako će se izračunati konačna cijena karte, uzimajući u obzir specifične popuste ili dodatke za odabrani način plaćanja. Ovaj pristup omogućio je lakšu proširivost – dodavanje novog načina plaćanja zahtijeva samo implementaciju nove strategije, bez izmjena u postojećem kodu. Također, Strategy je povećao čitljivost i modularnost koda, omogućujući fokusiranje na pojedinačne funkcionalnosti bez miješanja logike plaćanja u glavnu poslovnu logiku. Na ovaj način osigurana je održivost i prilagodljivost sustava. |

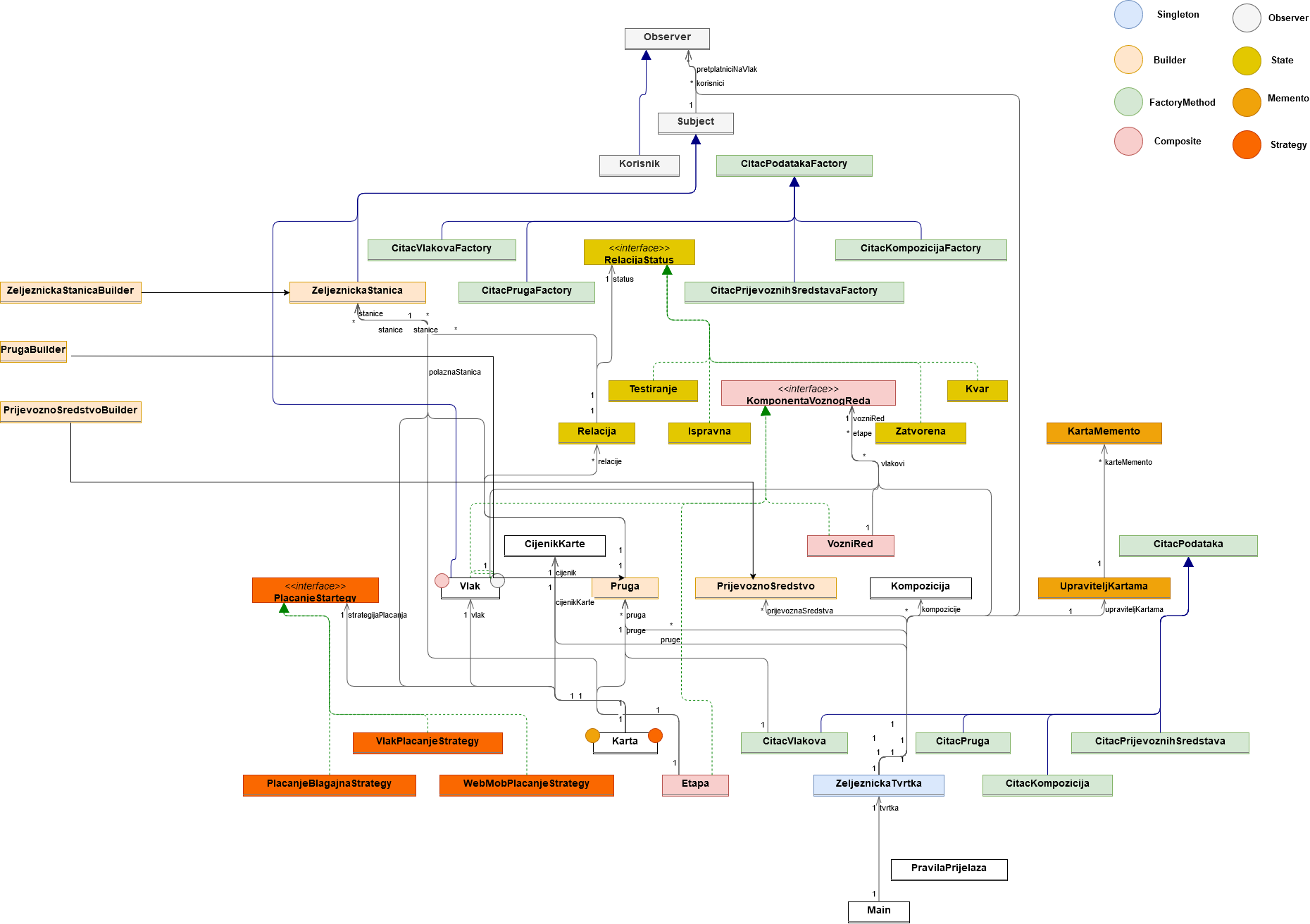
**Dio C.1. Opis promjena u odnosu na prethodnu zadaću**

1. Kreiran objekt CijenikKarte za spremanje informacija o cijenama karte
2. Kreirana komanda CVP
3. Kreirano sučelje za Strategy i sukladno tome kreirana 3 Strategy-a za plaćanje: PlacanjeBlagajnaStrategy, VlakPlacanjeStrategy, WebMobPlacanjeStrategy
4. Kreiran objekt Karta sa potrebnim informacijama i Strategy-om
5. Kreiran UpraviteljKartama i KartaMemento za spremanje informcija o kartama
6. Kreirane komande za kreiranje, karta, spremanje i dohvaćanje informacija o njima: KKPV2S, IKKPV, UKP2S
7. Kreiran objekt Relacija te je u objekt Pruga dodano da sadrži listu Relacija
8. Kreirano sučelje RelacijaStatus i 4 statusa koja implementiraju sučelje: Ispravna, Kvar, Zatvorna, Testiranje. Kreirana klasa PravilaPrijelaza kako bi se znao prijelaz iz pojedinog stanja
9. Kreiran objekt Relacija koji sadrži RelacijaStatus. Dodano da Pruga sadrži listu relacija te kreirana ograničenja nad kreiranjem Relacija.
10. Implemetirane komande za kreiranje, izmjene i ispis relacija: PSP2S, IRPS
11. Dodana ograničenja za kupovinu karata po neispravnim relacijama (komanda KKPV2S)

**Dio C.2. Opis funkcionalnosti za uzorak dizajna Command**

-Nije implementirano

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

****

1. N – dodan u 2. zadaći, P – promijenjen u 2. zadaći, S – bez promjena u 2. zadaći [↑](#footnote-ref-1)
2. N – dodan u 2. zadaći, P – promijenjen u 2. zadaći, S – bez promjena u 2. zadaći [↑](#footnote-ref-2)