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

Obrazac za zadaću na predmetu "**Uzorci dizajna**" ak. god. 2022./2023.

Ime i prezime studenta/ice: Alan Vranić

Matični broj: 0016130344

Dio A. Osnovni podaci o zadaći

R.br.	Pitanje	Odgovor	
1.	Grupa na seminaru:	G2	
2.	Broj i naziv zadaće:	3	Završena brodska luka
3.	Procjena vremena za realizaciju bez decimala):	25 sati	
4.	Procjena % završenosti (bez decimala):	100% / 100%	
5.	Procjena bodova za izradu zadaće (1 decimala):	15 / (DZ3 - 15)	
6.	Žalim prezentirati zadaću:	NEMA PREZENTACIJE	
7.	Koji dijelovi iz opisa zadaće nisu realizirani:	Svi dijelovi iz opisa su realizirani.	
8.	Postoji li dio zadaće koji vrijedi posebno istaknuti i zašto:	Cijela zadaća od početka do kraja pisana je izuzetno dosljednom i preglednom strukturom i stilom koda. MVC uzorak je jasno segregiran u svoje pripadne klase. Uvedeno je 12 različitih uzoraka na mjestima gdje oni imaju smisla. 100%-tna funkcionalnost od sve tri zadaće je ostvarena u potpunosti, i sve to u ispod 2200 linija koda.	
9.	Postoje li dijelovi zadaće koji imaju pogrešku u radu i koje:	Prema mojem opsežnom testiranju, niti jedan dio programa ne sadrži pogreške u radu.	
10.	Da li ste koristili tuđi programski kod u realizaciji zadaće izvan spomenutih izvora na nastavi:	Ne	
11.	Da li ste koristili programska rješenja ili dijelove programskog koda od drugih kolega:	Ne	

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

Naziv uzorka dizajna	Klase koje sudjeluju u uzorku dizajna	Opis razloga odabira uzorka dizajna
Singleton	BrodskaLuka, VirtualniSat	Brodska luka ponaša se kao baza podataka sa svim učitanim brodovima, vezovima itd. Na sličan način, VirtualniSat pohranjuje trenutno vrijeme. Svi podaci, kao i virtualno vrijeme moraju biti dostupni na više mjesta kroz aplikaciju, stoga je ovo savršena primjena za Singleton ozorak dizajna. Također, potrebno je da postoji samo jedna brodska luka i samo jedan virtualni sat, stoga ovaj uzorak rješava i taj problem.
Proxy	IServis, Servis, ServisProxy	Proxy služi za više stvari, no u ovoj zadaći je primarni cilj iskoristiti proxy kao posrednika koji prije izvršavanja svake naredbe ispisuje virtualno vrijeme. Ponekad nakon neke naredbe ispiše i dodatne potvrdne podatke. U budućim zadaćama, proxy bi se lako mogao proširiti da bilježi ostale greške, vrši autentifikaciju ili validira ulazne podatke prije nego one dođu do naredbe koja se želi izvršiti. Također, proxy započinje rad virtualnog sata čim je instanciran.
Factory method	IParametriZauzetosti, IZauzetost, ParametriRasporedneZauzetosti, ParametriRezervacijskeZauzetosti, RasporednaZauzetost, RasporednaZauzetostKreator, RezervacijskaZauzetost, RezervacijskaZauzetostKreator, ZauzetostKreator	U 1. zadaći to je najveći uzorak dizajna i po meni najkorisniji. U zadaći se zauzetost može predočiti na dva načina – putem periodičnog rasporeda i putem zahtjeva rezervacije koji su nepredvidljivi. Ipak, oboje predstavljaju isti koncept – zauzeti period vremena na pojedinom vezu za pojedini brod. Odlučio sam koristiti Factory method kako bi olakšao instanciranje tih dviju klasa (putem zasebnih kreatora), te dodatno napravio sučelje IZauzetost koje ima definiranu metodu ZauzetURasponu. Na taj način, raspored i zahtjev rezervacije definira na svoj način što i kada znači biti zauzet u nekom rasponu datuma, a korištenje tih klasa je iznimno jednostavno te je uvelike smanjilo količinu koda koju sam morao pisati, što se uvelike reflektiralo na kvalitetu zadaće. Također je uveden interface IParametriZauzetosti kako bi mogao instancirati klase IZauzetosti koje imaju različite argumente za konstruktor, a svejedno se pridržavaju svih pravila strukture ovog uzorka.

Dio B.2. Dokumentacija rješenja 2. zadaće

Naziv uzorka dizajna	Klase koje sudjeluju u uzorku dizajna	Status[1]	Opis razloga odabira uzorka dizajna
Singleton	BrodskaLuka	S	Navedeno u tablici iznad (nepromijenjeno)
Singleton	VirtualniSat	S	Navedeno u tablici iznad (nepromijenjeno)
Singleton	VirtualniSat	N	Uvijek postoji samo jedan ispisivač tablice jer se u njemu prilikom učitavanja spremaju opcije prikaza tablice. Također, te opcije se mogu promijeniti tijekom izvršavanja programa. Na ovaj način, sve vezano uz ispis tablica i njihovo formatiranje je na jednom mjestu dostupnom kroz cijelu aplikaciju.
Proxy	Navedeno u tablici iznad (nepromijenjeno)	S	Navedeno u tablici iznad (nepromijenjeno)
Factory method	Navedeno u tablici iznad (nepromijenjeno)	S	Navedeno u tablici iznad (nepromijenjeno)
Template method	BrodCsvCitac, BrodskaLukaCsvCitac, CsvCitacAbstract, KanalCsvCitac, MolCsvCitac, MolVezCsvCitac, RasporedCsvCitac, VezCsvCitac, ZahtjevRezervacijeCsvCitac	N	Primjena Template method uzorka je idealna u slučajevima kad imamo algoritam u kojemu je potrebno izmijeniti ili imati varijacije pojedinih koraka. U ovom slučaju, proces učitavanja CSV datoteka je 90% sličan. Jedini korak koji varira je korak parsiranja podataka, koju konkretne implementacije CSV čitača mogu naslijediti i nadjačati. Na ovaj način, dolazi do bolje iskorištenosti koda (DRY), i svaki čitač je zaslužan samo za svoj dio.
Chain of command	HitniPrijevoz, IHitniPrijevozHandler	N	Funkcionalnost, način implementacije i razlog odabira uzorka Chain of Command dostupan je ispod, u dijelu Dio C.2.
Visitor	IVezVisitor, IZauzetiVez, VezZauzetostPremaVrstamaVisitor	N	Uzorak dizajna Visitor implementiran je jer je zahtjevan u zadaći. Visitor nam omogućava da odvajamo algoritme od objekata nad kojim oni operiraju. U našem slučaju, primjenom Visitora omogućili smo sumarno zbrajanje zauzetih vezova prema vrstama za određeno vrijeme. Visitor posjećuje objekte koji implementiraju sučelje IZauzetiVez što im dozvoljava da „prihvate“ Visitora, zbog čega on može pristupiti ključnim podacima koje koristimo za sumarno zbrajanje.
Mediator	IMediator, ISubscriber, OdgovorLuckeKapetanije, PorukaVhfKanala,	N	Mediator je najključniji uzorak dizajna u 2. zadaći. Uzorak dizajna Mediatora omogućuje nam da se Kanali ponašaju kao

[1] N – dodan u 2. zadaći, P – promijenjen u 2. zadaći, S – bez promjena u 2. zadaći

	ZahtjevZaOdjavu, ZahtjevZaPrivezNaRezerviraniVez, ZahtjevZaPrivezNaSlobodanVez		posrednici između svih korisnika na toj frekvenciji. Na ovaj način, brodovi i brodska luka mogu si međusobno slati poruke, bez da direktno znaju jedan za drugog. Brodska luka uvijek sluša na svim frekvencijama. Cilj je da pojedini brod i brodska luka mogu komunicirati, no svi ostali posrednici će također dobiti te poruke. Bitno je napomenuti da u ovom slučaju Mediator ima i ulogu uzorka Observer. Odnosno, nakon što brod pošalje poruku Mediatoru (kanalu), on će obavijestiti sve svoje pretplatnike da je stigla poruka, te istu proslijediti. Svi brodovi i brodska luka odlučuju što će raditi sa dobivenim porukama, što stvara veliku modularnost.

Dio B.3. Dokumentacija rješenja 3. zadaće

Naziv uzorka dizajna	Klase koje sudjeluju u uzorku dizajna	Status[2]	Opis razloga odabira uzorka dizajna
Singleton	BrodskaLuka	S	Ispisom podataka sada upravlja MVC.
Singleton	VirtualniSat	S	Navedeno u tablici iznad (nepromijenjeno)
Singleton	IspisivacTablice	P	Ispisom podataka sada upravlja MVC.
Proxy	Navedeno u tablici iznad (nepromijenjeno)	S	Navedeno u tablici iznad (nepromijenjeno)
Factory method	Navedeno u tablici iznad (nepromijenjeno)	S	Navedeno u tablici iznad (nepromijenjeno)
Template method	Navedeno u tablici iznad (nepromijenjeno)	S	Navedeno u tablici iznad (nepromijenjeno)
Chain of command	Navedeno u tablici iznad (nepromijenjeno)	P	Ispisom podataka sada upravlja MVC.
Visitor	Navedeno u tablici iznad (nepromijenjeno)	P	Ispisom podataka sada upravlja MVC.
Mediator	Navedeno u tablici iznad (nepromijenjeno)	P	Ispisom podataka sada upravlja MVC.
Composite	BrodskaLuka, Component, Composite, Leaf, Mol, Vez	N	U opisu zadaće je zadano da je potrebno implementirati uzorak Composite. Composite je koristan kada imamo strukturu (kao u našem slučaju) koja se može prikazati strukturom stabla. U našem slučaju, struktura je Brodska luka -> Mol -> Vez. Brodska luka je glavni Composite, a mol je njegov Leaf. Također, Mol je composite a Vez je njegov leaf. Na ovaj način ostvarujemo dinamičnu strukturu koju je lako proširiti i iterirati.
Iterator	IIterator, IMolIteratorCreator, IVezIteratorCreator, MolIterator, VezIterator	N	U opisu zadaće je zadano da je potrebno implementirati uzorak Iterator. Nadovezujući se na uzorak Composite, uvodimo uzorak Iterator kako bi se podaci brodske luke, molova i vezova mogli iterirati i ispisati, a bez da se otkriva njihova unutarnja struktura. Brodska luka može stvoriti iterator molova i vezova, dok mol može stvoriti iterator molova.
Memento	IMemento, IOriginator, StanjeZauzetostiMemento, BrodskaLuka	N	Funkcionalnost spremanja i ponovnog učitavanja ostvarena je korištenjem uzorka Memento. Memento omogućuje fleksibilno spremanje podataka na način da su samo originatori dostupni svi podaci, a ostali elementi pristupaju mementu isključivo preko sučelja te se na taj način očuva učahuriranje i privatnost podataka.
Model-view-controller	Model, View, Controller, IModel	N	U opisu zadaće je zadano da je potrebno implementirati uzorak MVC. MVC je najveći uzorak u zadaći 3 te je zahtjevao kompletno restrukturiranje cijelog programa. U principu, ono što je prije bio Servis (glavni dio logike programa), sada se zove Model.

[2] N – dodan u 2. zadaći, P – promijenjen u 2. zadaći, S – bez promjena u 2. zadaći

			Cijeli unos podataka sada obrađuje kontroler, koji naređuje modelu da vrši operacije nad strukturom podataka. Nakon što se naredbe izvrše, model obaviještava view da je došlo do promjena, te je view zadužen da ispiše sve potrebne podatke. Model, view i kontroler rade u paru kako bi se ostvarila separacija odgovornosti, te na ovaj način možemo proširiti program i točno znati koji je dio odgovoran za koju funkcionalnost.
Observer	IObserver, Observable, Model, View	N	Observer je uzorak koji je implementiran jer je interno korišten u uzorku MVC. Konkretno, model nasljeđuje od klase Observable, te se na njega mogu pretplatiti Observeri koji će reagirati na određene promjene. Konkretno, view se pretplaćuje na model tako da može ispisati izmijenje podatke nakon što model izvrši naredbu koju mu je proslijedio kontroler. Na ovaj način, ne dolazi do uske povezanosti između dijelova programa, što je dobra stvar za održivost i modularnost koda.

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

Većina je promjena opisana u tablici iznad. Bitno je samo napomenuti kako je skoro svaki dio programa (pogotovo na mjestima gdje se ispisuju ili unose podaci) kompletno izmijenjen kako bi se mogao implementirati MVC uzorak dizajna.

Također, sam ispis implementiran je pomoću VT100 ESC kodova za direktno pozicioniranje te je kompletno fleksibilan i nudi mnogo kombinacija postavki za što bolji rad.

Dio C.2. Opis funkcionalnosti za uzorak dizajna Chain of Resposibility

Uzorak dizajna Chain of responsibility u 3. je zadaći ostao nepromijenjen, osim što njegovim ispisom podataka sada upravlja uzorak MVC.

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