**Manager Karti za letove**

Manager karti za letove je Java Swing aplikacija, koja za cilj ima pružiti korisnicima mogućnost kupovine karti, pregledom i upravljanjem istih. Aplikacija omogućava administratorima istu funkciju, alii m je omogućen i unos zrakoplova, kao i unos letova, čije su karte dostupne za kupovinu.

U okviru projekta sam ispunio osnovne smjernice koje su se tražile od nas, poput upravljanja nezavisnostima koristeći Maven, osiguravajući rad s podatkovnom bazom, spremanje datoteka u MVC predlošku.

Od ostalih predložaka korišteni su:

**Command predložak**

Command predložak sam koristio u korisničkom okviru (slika okvira je prikazana dalje u dokumentaciji) kako bih omogućio korisniku (kupcu karte) filtriranje dostupnih letova prema određenim uvjetima. Filteri su:

* Najstarije prvo – prikazuje letove s najstarijim datumom leta prema najnovijim (letovi u 2023. prije letova u 2024. godini)
* Najnovije prvo – prikazuje letove s najnovijim datumom leta prema najstarijim (letovi u 2025. prije letova u 2023. godini)
* Postoji li prva klasa – prikazuje samo letove koji imaju prvu klasu sjedala\*
* Postoji li poslovna klasa – prikazuje samo letove koji imaju poslovnu klasu sjedala\*
* Postoji li ekonomična klasa – prikazuje samo letove koji imaju ekonomičnu klasu sjedala\*

U klasičnoj izvedbi Command predloška omogućen je *redo*  i *undo* filtera odabirom tih opcija unutar JMenuBar elementa okvira, ili koristeći odgovarajućeg prećaca.

(\*Ukoliko je jedan od ovih filtera ukljućen, npr. prva klasa, biti će prikazani svi letovi koji imaju dostupnu prvu klasu, čak i ako imaju i poslovnu i/ili ekonomičnu dostupnu također)

**Observer predložak**

Observer predložak je također korišten u korisničkom okviru. On je zadužen za prikaz kratkog teksta koji prikazuje podatke o odabranom letu u zato predviđenom JTextArea elementu. Pošto sam omogućio JTable element MouseEventListener, da korisnik može odabrati let klikom po tablici, on je Observable objekt. Dok je JTextArea Observer I mijenja svoje stanje ovisno o odabiru korisnika. (Slika priložena u nastavku)

**Strategy predložak**

Na predavanju smo ovaj predložak koristili kada korisnik odluči spremiti (ili učitati) podatke koristeći JFileExplorer gdje bi Strategy predložak dobio informaciju želi li korisnik upravljati datotekom u obliku .bin ili .txt i tom informacijom odabrao koju klasu (strategiju) će koristiti. Međutim, ja u svojoj aplikaciji nisam vidio mogućnost ovakvog pristupa, pa sam odlučio koristiti Strategy predložak koji za informaciju dobiva ActionCommand pritisnutog JButton elementa i time bira koju strategiju će pozvati. U ovom slučaju Strategy predložak je zadužen za pozivanje odgovarajućeg načina za interakciju s bazom podataka.

**Singleton predložak**

Singleton predložak je jednostavan predložak dizajna u Java programerskom jeziku. Omogućuje instanciranje objekta jednom, tako da možemo pozivati samo refrencu na taj objekt gdje god nam treba, umjesto da instanciramo novi objekt kad god ga trebamo koristiti. Potrebno je napraviti konstruktor objekta s privatnim modifikatorom pristupa i onda ga statički instancirati. Ovaj predložak sam koristio za Kontroler klasu, da ga mogu instancirati samo jednom jer je potreban unutar gotovo svakog okvira, i AuxCLS klasu koja je zadužena za spremanje podataka koji su korišteni dalje u kodu.

**SOLID princip**

**Single responsibility** – Dio klasa se pridržava ovog principa iako se na nekoliko mjesta primjećuje klase obavljaju više od jednog zadatka (primarno u JPanel klasama)

**Open/Closed princip** - U izgradni aplikacije se nisam obazirao na ovaj dio SOLID principa. Open/closed princip je odličan za izbjegavanje bugova za aplikacije u produkciji, koji će dobivati podršku nakon izlaska da se smanji broj mogućih *bugova*.

**Liskov Substitution** -