Tema proiectului este **Masina inteligenta** si consta intr-o aplicatie Java, cu interfata de tipul Java Swing. Deoarece este o aplicatie de tip JavaFX, am folosit modelul de proiectare MVC. In aplicatie am implementat design pattern-urile: *State, Observer si Command.*

In clasa Main are loc binding-ul dintre controlerele folosite in interfata si metodele specifice care se executa. In clasa Controller este descrisa logica de implementare a raspunsurilor in functie de interactiunea in cadrul interfetei.

Design Pattern-ul *State* a fost implementat pentru a acoperi cerinta urmatoare: „Comportamentul mașinii este reprezentat printr-un set de stări.”. Am folosit 2 stari ale masinii: Oprita si Pornita. Acestea sunt descrise in enum-ului Stari si prin clasele **StareOprita** si **StarePornita**. Clasele **StareOprita** si **StarePornita** implementeaza interfata **IStare** si suprascriu metodele din aceasta: **pornesteMotorul(); dezactiveazaFranaDeMana(); introduInViteze(); porneste(); activareSenzoriParcare(); detectareLoc(); centrarePeLoc(); ocupareLoc(); oprireMotor().** Clasele specific acestui design pattern sunt grupate in pachetul **state**.

In clasa **MasinaInteligenta** am declarant un obiect de tipul **IStare**, cu referinta caruia sa gestionez starea curenta, respective metodele specifice starii curente.

Design Pattern-ul Observer este implementat corespunzator cerintei: “Mașina poate să notifice utilizatorii înregistrați despre evenimentele care apar pe parcursul funcționării.”. Pachetul **observer** contine clasele specific acestui design pattern, respectiv doua interfete **IObserver** si **ISubject** si o clasa **Utilizator**. Clasa **Utilizator** implementeaza interfata **IObserver** si suprascrie metoda **receptionareMesaj(String mesaj)** pentru definirea modalitatii de notiicare a utilizatorilor abonati la masina inteligenta.

Clasa **MasinaInteligenta** implementeaza interfata **ISubject** si suprascrie metodele **adaugaObserver**() si **stergeObserver**() pentru a gestiona lista de observatori abonati la masina inteligenta si metoda **notificaMesaj**(**String mesaj**) care notifica utilizatorii din lista abonatilor in functie de evenimentele care se produc (schimbare stare masina, abonarea/dezabonarea unui utilizator si comenzile efectuate). La abonarea unui utilizator nou, se apeleaza metoda **adaugaObserver**() si se primeste prin intermediul metodei **notificaMesaj**(), mesajul: Utilizatorul „nume utilizator” s-a abonat, respectiv la dezabonare, se apeleaza metoda **stergeObserver**() si se primeste tot prin intermediul metodei **notificaMesaj**(), mesajul: Utilizatorul „nume utilizator” a fost dezabonat”. Aceste mesaje se primesc de catre toti utilizatorii abonati la masina inteligenta.

Design Pattern-ul Command este implementat pentru a acoperi cerinta: “Utilizatorii interacționează cu mașina prin intermediul comenzilor.” Obiectele specifice folosite in implementare se afla in pachetul **command**. S-au definit doua clase **ComandaPlecare** si **ComandaParcare** care implementeaza metoda **executa**(), metoda care se ocupa cu controlul **Receiver**-ului (localizata in interfata **ICommand**). Aceste clase reprezinta legatura dintre **Receiver** si actiunile specifice comenzilor. Interfata IReceiver contine actiunile si se ocupa cu efectuarea acestora. Actiunile sunt **pornesteMotorul**(); **dezactiveazaFranaDeMana**(); **introduInViteze**(); **porneste**(); **activareSenzoriParcare**(); **detectareLoc**(); **centrarePeLoc**(); **ocupareLoc**(); **oprireMotor**().

Clasa **MasinaInteligenta** creeaza comanda si cere efectuarea actiunilor specifice acesteia.

In interfata cu utilizatorul, starile Oprita si Pornita sunt descrise de campul „Stare Masina” care are doua radio button-uri specifice. Utilizatorul curent este ales in campul „Utilizator curent”, prin selectarea unui radio button specific utilizatorului (Utilizator1 sau Utilizator2). Check box-ul Abonat Masina este specific fiecarui utilizator si exemplifica abonarea s-a la masina inteligenta pentru a putea executa comenzi si a primi notificari despre actiunile celuilalt utilizator abonat. Actiunile definite sunt grupate in doua comenzi si se executa prin apasarea unuia dintre butoanele: Executa Plecare si Executa Parcare. Sub aceste butoane se afla o casuta de text, in care se afiseaza notificarile pentru utilizatorii abonati si executarea actiunilor conform starii selectate.

Realizat de Stroia Madalina Paula, an 2, master E-Business