1. **Definirea si detalierea pattern-urilor utilizate (minim 3)**

Intr-un ParcAuto au loc zilnic transferuri de vehicule, insemnand achizitionarea de catre parc de noi vehicule, respectiv vanzarea acestora.

Definirea unui anumit vehicul este realizata prin design patternul creational **FactoryMethod.**

Fiind un orasel de provincie, primarul a dorit ca acesta sa fie singurul parc auto, cerinta obtinuta prin implementarea design patternului **Singleton**.

Parcul auto de asemenea permite vanzarea de pachete promotionale, ce pot fi constituite din mai multe vehicule sau alte pachete promotionale, lucru obtinut prin patternul structural **Composite.**

1. **Definirea si detalierea metodelor testate prin Unit Testing (minim 10 metode) si Definirea si descrierea Test Case-urilor**

Am testat corectitudinea metodelor din clasele:

* + PachetPromotional: adauga(), sterge()
  + ParcAuto: automatizare fabricare vehicule (citirea datelor din fisier pentru testare fabricare vehicule, avand input correct); am creat si metodele setUpGlobal() si tearDownGlobal() pentru instantiere Fabrica. Deschiderea si inchiderea fisierului se face in metodele setUp(), respectiv tearDown(), apelate inaintea rularii fiecarui unit test.
  + ProprietatiVehicul: testare valori aberante pentru anul fabricatiei (valori negative si mai mari decat anul current)
  + Testare implementare pattern Singleton
  + Vehicul: testare constructor cu valoarea “null”, setare pret valori aberante (negative, subunitar, < 100 um), testare rezultat metoda getNrRori()
  + Mockito: testare creare pachet promotional si vanzare vehicul folosind Mock Objects

1. **Definirea si descrierea Test Suite-ului**
   * La rulare TestSuite se vor rula defapt toate testcaseurile si unittestele adaugate in colectie. Pentru adaugarea doar a anumitor metode se implementeaza in acele testcase-uri constructorul care sa primeasca numele metodei ca parametru