Cerințe obligatorii

1. Pattern-urile implementate trebuie sa respecte definiția din GoF discutată în cadrul cursurilor și laboratoarelor. Nu sunt acceptate variații sau implementări incomplete.
2. Pattern-ul trebuie implementat corect în totalitate pentru a fi luat în calcul.
3. Soluția nu conține erori de compilare.
4. Pattern-urile pot fi tratate distinct sau pot fi implementate pe același set de clase.
5. Implementările care nu au legătura funcțională cu cerințele din subiect NU vor fi luate în calcul (preluare unui exemplu din alte surse nu va fi punctată).
6. NU este permisă modificare claselor/interfetelor primite.
7. Soluțiile vor fi verificate încrucișat folosind MOSS. Nu este permisă partajarea de cod între studenți. Soluțiile care au un grad de similitudine mai mare de 30% vor fi anulate.

Cerințe Clean Code obligatorii (soluția este depunctata cu câte 2 puncte pentru fiecare cerința ce nu este respectată) - maxim se pot pierde 4 puncte

1. Pentru denumirea claselor, funcțiilor, testelor unitare, atributelor și a variabilelor se respecta convenția de nume de tip Java Mix CamelCase.
2. Pattern-urile și clasa ce conține metoda main() sunt definite in pachete distincte ce au forma *cts.nume.prenume.gGrupa.pattern.model*, *cts.nume.prenume.Grupa.pattern.main* (studenții din anul suplimentar trec “as” în loc de gGrupa).
3. Clasele și metodele sunt implementate respectând principiile KISS, DRY și SOLID (atenție la DIP).
4. Denumirile de clase, metode, variabile, precum și mesajele afișate la consola trebuie sa aibă legătura cu subiectul primit (nu sunt acceptate denumiri generice). Funcțional, metodele vor afișa mesaje la consola care sa simuleze acțiunea cerută sau vor implementa prelucrări simple.

Se dezvoltă o aplicație software destinată unui magazin de Biciclete.

4p. Pentru un magazin care produce și vinde Biciclete se dorește implementarea unei aplicații care să ajute la gestionarea fabricării de biciclete. Astfel în cadrul aplicației inginerii au posibilitatea creării obiectelor pentru seturi de biciclete care au mai multe caracteristici, dintre care unele sunt opționale: diametruRoti, tipFrana, cascaProtectie, ochelari, etc. Orice set creat nu mai poate fi modificat urmand sa fie vandut asa cum a fost creat. Să se implementeze modulul care îi va ajuta pe gestionari în procesul de creare al obiectelor de tip Bicicleta. Clasa Bicicleta trebuie sa implementeze interfata IBicicleta.

1p. Să se testeze soluția prin crearea a cel puțin patru obiecte prin intermediul modulului implementat

4p. Soluția trebuie să permită crearea de Biciclete de diferite tipuri: MTB, Electrica, Trekking. Pentru fiecare tip de bicicleta este folosită o clasă aferentă tipului de bicicletă care implementează interfața IBicicleta. Să se implementeze modulul care îi va ajuta pe inginerii magazinului producator de biciclete în procesul de creare al obiectelor din familia IBicicleta..

1p. Să se testeze soluția prin crearea a cel puțin patru obiecte din cel puțin două categorii diferite din familia bicicletelor.