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 activității unui magazin online.

4p. Magazinul online dorește dorește să comercializeze o gamă largă de produse prin intermediul platformei. Astfel sunt luate în considerare Produse Perisabile, Produse Electronice și Produse Bio. Soluția trebuie să permită generarea facilă de instanțe din această familie de clase. Se ia în calcul că toate produsele au în comun interfața ProdusGeneric.

1p. Să se testeze soluția prin generarea de obiecte diferite aparținând familiei de Produse (cel puțin un obiect pentru fiecare tip)

4p. În cadrul platformei trebuie avut grijă ca un produs să existe o singură dată, dar avem mai multe produse. Unicitatea se asigura la nivelul numelui produsului. În acest mod magazinul evită ca un produs să apară de mai multe ori pe platformă. În cazul în care se dorește adăugarea unui produs care există deja pe platformă se va modifica stocul acestuia. Produsl trebuie să implementeze interfața ProdusGeneric.

1p. Să se testeze soluția prin crearea a cel puțin patru obiecte pentru prezentarea avantajelor design pattern-ului implementat.