Observații

1. Implementările de Design Patterns trebuie să fie conform definiției GoF discutată în cadrul cursului și laboratoarelor. Implementările incomplete sau variațiile non-GoF ale patternurilor nu sunt acceptate și nu vor fi punctate.
2. Sunt luate în considerare doar implementările complete și corecte, implementările parțiale nu se punctează.
3. Soluțiile ce conțin erori de compilare nu vor fi evaluate.
4. Patternurile pot fi implementate separat sau utilizând același set de clase.
5. Implementările generale de design patterns, care nu sunt adaptate cerinței și nu rezolvă problema cerută nu vor fi luate în considerare.
6. Clasele/interfețele primite nu pot fi modificate.
7. Soluțiile vor fi verificate cu software antiplagiat. Partajarea de cod sursă între studenți nu este permisă. Soluțiile cu un nivel de similitudine peste 30% vor fi anulate.
8. Se acordă 1 punct din oficiu.

Cerințe Clean Code (nerespectarea lor va duce la depunctarea cu 2 puncte pentru fiecare cerință) - se pot pierde 8 puncte în total

1. Clasele, funcțiile, atributele și variabilele vor fi denumite conform convenției Java Mix CamelCase.
2. Fiecare pattern precum și clasa ce conține metoda `main()` vor fi definite în pachete diferite de forma `cts.nume.prenume.g<numarul_grupei>.pattern.<denumire_pattern>`, respectiv `cts.nume.prenume.g<numarul_grupei>.main` (studenții de la recuperare vor utiliza „recuperare” în loc de numărul grupei).
3. Clasele și metodele nu trebuie să încalce principiile KISS, DRY, YAGNI sau SOLID.
4. Numele claselor, metodelor, variabilelor și mesajele afișate la consolă trebuie să fie strict legate de subiectul curent (numele generice nu sunt acceptate). Mesajele afișate trebuie să simuleze scenariul cerut.
5. Nu sunt permise „magic numbers” sau valori hardcodate. Formatarea codului sursă trebuie să fie cea standard.

3p. Ați creat o aplicație mobilă ce vă permite să transmiteți live evenimente sportive. Realizați faptul că cu cât evenimentul este transmis live pe mai multe platforme, cu atât durează mai mult să porniți live-ul, cu toate că feed-ul live este același. Știind că o transmisiune live implementează interfața `Live`, utilizați un design pattern ce rezolvă problema de performanță. Țineți cont și de faptul că odată create, transmisiunile live pot avea detalii diferite (exemplu: altă listă de comentarii).

1.5p. Testați implementarea prin crearea a 3 transmisiuni live aferente aceluiași eveniment sportiv pe 3 platforme de streaming diferite. Demonstrați faptul că adăugarea unui comentariu pe una dintre platforme nu va face ca acesta să apară pe altă platformă.

3p. Aplicația permite transmisiuni live pentru 3 sporturi diferite: fotbal, baschet și handbal. Utilizați un design pattern ce permite crearea transmisiunii în funcție de tipul de sport știind că toate tipurile de sport derivează clasa abstractă `Sport` și că sportul este ales la momentul execuției programului împreună cu numele echipelor (echipa gazdă și echipa oaspete). Luați în calcul și faptul că pe viitor se dorește implementarea altor categorii de sporturi, iar acest lucru nu trebuie să afecteze/modifice implementarea curentă.

1.5p. Testați în main implementarea prin crearea a minim 3 transmisiuni pentru sporturi diferite. Utilizați o modalitate de evita referințele la clase concrete în metoda principală.