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 corect 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 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 1 punct 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 respectă convenția de nume de tip Java Mix CamelCase;
2. Pattern-urile și clasa ce conține metoda main() sunt definite în pachete distincte ce au forma *cts.nume.prenume.gNrGrupa.denumire_pattern*, *cts.nume.prenume.gNrGrupa.main* (studenții din anul suplimentar trec “as” în loc de gNrGrupa)
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 legătură 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 pentru o companie care asamblează și comercializează sisteme complete de gaming.

4p. Aplicația implementată trebuie să permită crearea diferitelor configurații ale sistemelor de gaming, în funcție de cerințele clienților. Pe lângă elementele care nu pot lipsi dintr-un astfel de sistem: placă de bază, memorie RAM, stocare ssd și placă video speciale pentru gaming, sistemele pot conține și multe alte echipamente periferice și accesorii, cum ar fi: mouse, tastatură, cameră web, căști, scaun de gaming etc.. O configurație creată este finală și nu mai poate fi modificată. Implementarea se va efectua plecând de la interfața **IGamingSystem**.

```
public interface IGamingSystem {
    public void printConfiguration();
}
```

1p. În main() trebuie să se implementeze o secvență de cod relevantă pentru evidențierea design pattern-ului implementat.

După asamblare, sistemele trebuie testate pentru a verifica eventualele probleme de compatibilitate, instalare, defecte etc.. Compania a achiziționat un sistem hardware specializat pentru testare, pe care un modul de testare ce va fi implementat în aplicația curentă îl va folosi în procesul de testare.

4p. Se va crea în cadrul aplicației implementate un modul de testare. Din motive de licențiere, odată creată o componentă de testare, aceasta va fi utilizată pentru testarea tuturor sistemelor configurate. Nu va putea fi creată o altă entitate de testare. Implementarea va fi realizată prin implementarea interfeței **ITesingModule.**

```
public interface ITesingModule{
// metoda va returna "rezultatele testării"
    public String test();
}
```

1p. În main() trebuie să se implementeze o secvență de cod relevantă pentru evidențierea design pattern-ului implementat.