S2 – Abonamente

Cerințe obligatorii

1. Pattern-urile implementate trebuie să respecte definiția din GoF discutată în cadrul cursurilor și laboratoarelor. Nu sunt acceptate variații sau implementări incomplete.
2. Pattern-ul trebuie implementat corect și complet (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 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) - se pot pierde maxim 8 puncte

1. Pentru denumirea claselor, funcțiilor, atributelor și a variabilelor se respecta convenția de nume de tip Java Mix CamelCase discutată;
2. Pattern-urile și clasa ce conține metoda main() sunt definite in 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 legătura cu subiectul primit (nu sunt acceptate denumiri generice). Funcțional, metodele vor afișa mesaje la consola care să simuleze acțiunea cerută sau vor implementa prelucrări simple.

Se dezvoltă o aplicație software destinată unei companii de telecomunicații – telefonie mobilă.

5p. O companie care oferă servicii de telefonie mobilă pune la dispoziția clienților 3 tipuri de abonamente: **MobilityX** (cuprinde doar servicii de voce), **MobilityY** (cuprinde doar servicii de date), **MobilityZ** (cuprinde servicii de voce, de date și tv online). Fiecare tip de abonament pune la dispoziție o modalitate prin care se poate afla care va fi costul total cu acest abonament, în funcție de numărul de luni contractuale. Să se implementeze componenta aplicației care să poată crea abonamente la cerere, în funcție de tipul abonamentului cerut. Implementarea se va realiza plecând de la interfața **Subscription** atașată acestui enunț.

3p. Să se testeze soluția în metoda main() prin generarea de abonamente, astfel: se tastează numărul de abonamente ce urmează a fi generate și apoi se citesc de la tastatură detaliile care trebuie completate (citirea se realizează într-o buclă până la crearea numărului solicitat de abonamente).

9p. Compania își extinde activitatea prin crearea unei divizii separate care vinde clienților următoarele extra opțiuni: internet în roaming, minute în roaming, internet 5G. Aceste opțiuni se vând pe lângă beneficiile existente într-un abonament existent (la un abonament se pot atașa 1, 2 sau toate cele 3 extra opțiuni). Această nouă divizie acționează independent de celelalte structuri ale companiei iar accesul la abonamente se face pe baza unui API care furnizează abonamentele clasice, neexistând posibilitatea de modifica codul sursă existent pentru generarea abonamentelor cu extra opțiuni. Componenta implementată trebuie să expună o modalitate prin care se poate calcula costul total al abonamentului (abonament de bază + costuri extra opțiuni)

pentru un anumit număr de luni. Implementați o modalitate prin care să se genereze abonamente cu extra opțiuni, conform descrierii anterioare.

3p. Pattern-ul este testat în main() prin definirea și utilizarea a cel puțin un obiect din familia abonamentelor cu extra opțiuni. Se va exemplifica și calculul costurilor totale pentru o perioadă de timp.

```
public interface Subscription {
    /*

     */
    public float totalCosts(int durationInMonths);
}
```