S2 – Companie de asigurări

Cerințe obligatorii

1. Pattern-urile implementate trebuie să respecte definiția și structura discutate 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 asigurări.

5p. O companie de asigurări dorește implementarea unei aplicații prin care să țină evidența clienților care au adus pierderi mari companiei (sau care au fraudat compania de asigurări). Pentru a face asta, pentru fiecare client vrea să marcheze dacă acesta este Activ, Blocat sau DeAnalizat. Dacă este blocat, acesta nu mai poate încheia nicio poliță de asigurare. Dacă este în analiză va fi nevoie de o analiză suplimentară în momentul în care acesta dorește emiterea unei polițe. Orice client activ nu are restricții. Implementarea se va realiza plecând de la interfața **InsuranceCompany** atașată acestui enunț.

3p. Să se testeze soluția în metoda main() astfel încât să fie evidențiate toate cele 3 scenarii, câte unul pentru fiecare variantă posibilă.

9p. Compania de asigurări poate emite în momentul de față doar polițe standard (tot ceea ce cuprinde polița este dinainte stabilit și nu poate fi modificat).
Pentru a atrage mai mulți clienți, vor să extindă variantele de asigurai cu extra opțiuni: decontare directa, asistență VIP, suport tehnic, prioritate în soluționarea dosarelor de daună.
Extra opțiunile sunt acordate de un nou departament care nu poate emite asigurări de la zero, doar poate adaugă extra opțiuni, datele despre polițe fiind preluate de la celelalte departamente printr-un API (codul sursa care generează polițele nu este disponibil pentru utilizare in implementarea generării polițelor cu extra opțiuni). Implementarea se va realiza plecând de la interfața **InsurancePolicy** atașată acestui enunț.

S2 – Companie de asigurări

3p. Să se testeze soluția în metoda main() astfel încât să fie evidențiată implementarea cerințelor de mai sus.

```
public interface InsuranceCompany {
    public String issueInsurancePolicy();
}
```

```
public interface InsurancePolicy {

    public float getTotalCosts();
}
```