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.

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.

Realizați o aplicație software pentru o agenție de turism.

3p. Agenția deține o licență de turism unică ce trebuie imprimată pe toate documentele emise. Clasa aferentă licenței de turism trebuie să deriveze interfața `AbstractTourismLicense`. Metoda `setLicenseNumber()` trebuie să permită modificarea licenței o singură dată (orice viitor apel va arunca o excepție creată special în acest sens). Implementați un design pattern ce nu permite crearea mai multor licențe de turism simultan, ținând cont de faptul că aplicația este una multithreading.

2p. Testați implementarea prin crearea a 3 referințe la clasa creată, din care una într-un thread secundar. Demonstrați faptul că cele 3 variabile referă aceeași licență de turism.

3p. Agenția de turism vinde pachete turistice formate din: transport (cu avionul sau cu autocarul), cazare (durată, tip cameră, stele hotel, etc.) și activități extra (excursii de grup, vizite muzee, etc.). Clasa aferentă unui pachet turistic este derivată din interfața `AbstractHolidayPackage`. Agenția dorește să creeze pachete turistice în orice combinație (doar cazare, doar transport, transport + cazare, etc.) și în plus să nu permită modificarea unui pachet turistic odată ce a fost creat. Implementați un design pattern ce rezolvă situația descrisă.

2p. Testați în main implementarea prin crearea a minim 3 pachete turistice ce conțin diverse combinații de servicii. Demonstrați faptul că un pachet odată ce a fost creat nu mai poate fi modificat.