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 dronă autonomă de supraveghere a incendiilor de vegetație.

3p. Pentru a putea zbura, drona are nevoie să fie conectată în permanență la o baza de control, la un serviciu web cu detalii despre starea vremii și la un satelit GPS. Toate aceste conexiuni derivează interfața `ServiceConnection`. Implementați un design pattern ce permite ca aceste conexiuni, împreună cu alte conexiuni viitoare, să fie unice în cadrul aplicației software.

2p. Testați implementarea prin crearea celor 3 conexiuni. Demonstrați faptul că prin crearea unei noi conexiuni la unul dintre serviciile menționate anterior se obține de fapt o conexiune existentă.

3p. În timp ce zboară, drona emite rapoarte cu privire la situația curentă din teren. Toate aceste rapoarte derivează interfața `FieldReport`. Utilizați un design pattern ce permite ca în timpul zborului, pe baza situației din teren, drona să emită rapoarte de informare, de avertizare sau de alertare.

2p. Testați în main implementarea prin crearea a minim 4 rapoarte din diferite categorii.