**Cerințe obligatorii**

1. Pattern-urile implementate trebuie sa respecte definitia din GoF discutată în cadrul cursurilor și laboratoarelor. Nu sunt acceptate variații sau implementării incomplete.
2. Pattern-ul trebuie implementat corect în totalitate corect pentru a fi luat în calcul
3. Soluția nu conține erori de compilare
4. Testele unitare sunt considerate corecte doar dacă sunt implementate conform cerințelor și dacă metodele sunt corectate corespunzător pe baza lor
5. Pattern-urile pot fi tratate distinct sau pot fi implementate pe același set de clase

**Cerințe Clean Code obligatorii (soluția este depunctata cu câte 5 puncte pentru fiecare cerința ce nu este respectată**) - maxim se pot pierde 15 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, Test Case-urile, Excepțiile și clasa ce contine metoda main() sunt definite in pachete distincte ce au forma *cts.nume.prenume.gNrGrupa.teste*, *cts.nume.prenume.gNrGrupa.patternX*, *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 si atribute au semnificație (au legatura cu subiectul) în contextul subiectului. NU se folosesc denumiri generice/teoretice

Se dezvoltă o aplicație software destinată administrării unui cartier de blocuri.

**10p.** Dezvolți o soluție software pentru asociația de proprietari dintr-un bloc care sa permita gestiunea eficientă a evenimentelor ce pot apărea, evenimente care necesita luarea unor măsuri adecvate și care să protejeze viața locatarilor. Soluția procesează (conform interfeței *IProcesareEveniment*) diferite mesaje primite de la senzori/sisteme din bloc și va lua următoarele decizii

* dacă s-a declanșat alarma de incendiu atunci este notificat serviciul 112
* dacă este anunțată o inundație atunci este notificat administratorul
* dacă este doar o informare atunci sunt notificați locatarii
* (opțional) pentru alarmele de incendiu și inundație sunt anunțați și locatarii

**5p.** Pattern-ul este testat în main() prin exemplificarea scenariului în care sistemul procesează cel puțin unul dintre scenariile anunțate.

**5p.** Independent de implementarea scenariului anterior găsiți o soluție care să permite integrarea sistemului existent de procesare a alertelor (cu interfața *IProcesareEveniment*) în sistemul național de alertare ce este construit în jurul interfeței *IAlertarePublica*.

**5p.** Să se testeze soluția exemplificând utilizarea sistemului blocului în cadrul sistemului public de alertare.

**6p.** Dându-se clasa *Avion* și restricțiile definite prin comentarii să se implementeze teste unitare (toate testele unitare aferente unei metode sunt implementate într-un TestCase) care să cuprindă:

1. un unit test care să realizeze o testare *Boundary* pentru **setGreutateIncarcata()** **(1.5p)**
2. un unit test care să testeze o testare *Error Condition* pentru **setGreutateIncarcata().** Pentru test se va folosi o excepție custom **(1.5p)**
3. un unit test de tip *Existence* pentru ***constructorul cu parametri* (1.5p)**
4. un unit test de verificare de tip *Range* pentru ***constructorul cu parametri***;**(1.5p)**

**2p.** Să se implementeze o suită de teste care să conțină DOAR câte o metodă, la alegere, din fiecare test case

**2p.** Prin testele implementate sau prin adăugarea de teste noi sa se testeze *getPretBilet*() asigurând un code coverage de 100% pentru această metodă.