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 respecta conventia de nume de tip Java Mix CamelCase;
2. Pattern-urile, Test Case-urile, Excepțiile și clasa ce contine metoda main() sunt definite în 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)

Se dezvoltă o aplicație software necesară gestiunii transportului public dintr-un oraș.

10p. Orașul este împărțit pe zone de acces, astfel dacă un călător(Interfața ICalator) dorește să se deplaseze în aceeași zonă (maxim 5 km) folosește autobuzul, dacă dorește să călătorească în două zone (maxim 10 km) folosește tramvaiul, dacă dorește să călătorească în trei zone (maxim 15 km) folosește metroul, iar dacă vrea să călătorească în afara orașului folosește trenul. Să se implementeze modul care permite călătorilor să aleagă mijlocul de transport public adecvat.

5p. Pattern-ul este testat în main() prin definirea a cel puțin cinci călători care vor călători în zone diferite ale orașului.

5p. Călătorii în momentul de față pentru plata biletelor de transport folosesc carduri diferite pentru fiecare mijloc de transport deoarece acestea diferă între ele. Fiecare mod de transport este gestionat de o altă companie și fiecare are propriul soft. Trebuie implementat un modul pentru transportatorii cu metroul și tramvaiul care să permită utilizarea plății biletelor folosind cardul utilizat deja pentru plata biletelor în autobuze.

5p. Să se testeze soluția prin utilizarea aceluiași card de către un călător pentru plata a minimum două mijloace de transport public.

6p. Dându-se clasa *Facturare,* clasa *Produs* și următoarele restricții: o factură conține maxim 20 de produse, prețul unui produs este cuprins în intervalul [1,1000] să se implementeze teste unitare, gestionate în test case-uri diferite pentru fiecare metodă testată, care să cuprindă:

1. un unit test care să realizeze o testare *Range* pentru **setPret() (1.5p)**
2. un unit test care să testeze o testare *Exceptie* pentru **setPret();** dacă este nevoie de excepție se generează una de tip *ExceptiePretIncorect***(1.5p)**
3. un unit test de tip *Existence* pentru metoda ***calculValoareTVA();*** dacă este nevoie de excepție se generează una de tip *ExceptieFacturaFaraProduse* **(1.5p)**
4. un unit test de verificare de tip *CrossCheck* pentru metoda ***calculValoareTVA*****();(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 adaugarea de teste noi sa se testeze *setPret*() din clasa *Produs* asigurând un code coverage de 100% pentru această metodă.