Cerințe obligatorii

1. Pattern-urile implementate trebuie sa respecte definiția din GoF discutată în cadrul cursurilor și laboratoarelor. Nu sunt acceptate variații sau implementări incomplete.
2. Pattern-ul trebuie implementat în totalitate corect 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) - maxim se pot pierde 8 puncte

1. Pentru denumirea claselor, funcțiilor, testelor unitare, atributelor și a variabilelor se respecta convenția de nume de tip Java Mix CamelCase;
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 sa simuleze acțiunea cerută sau vor implementa prelucrări simple.

Se dezvoltă o aplicație software destinată unui restaurant.

3p. În cadrul aplicației de gestiune a gătirii preparatelor în cadrul unui restaurant, se dorește implementarea unui modul de asignare a unei liste de așteptare pentru preparatele care se doresc a fi gătite. Se cunoaște faptul că, în cadrul restaurantului, există un singur cuptor pe care bucătarii îl pot folosi pentru a găti preparatele. Pentru fiecare preparat se cunoaște timpul alocat gătirii lui precum și gradele la care trebuie gătit. Să se implementeze acest modul în context centralizat al managementului preparatelor asignate gătirii de către unicul cuptor.

1p. Să se testeze soluția prin crearea unui număr de 4 comenzi formate din preparate și să se asigneze într-o listă de așteptare la nivelul unicului cuptor al restaurantului.

4p. Cu timpul, s-a observat că existența unui unic cuptor întârzie foarte mult lansarea comenzilor. Astfel, restaurant a mai achizionat un număr de alte 3 cuptoare pentru a optimiza timpul de așteptare pentru gătirea preparatelor. Să se reorienteze implementarea curentă astfel încât bucătarii să aibă la dispoziție o listă de 4 cuptoare. Asignarea preparatelor la cuptoare se realizează printr-o minimizare a timpului de așteptare. De asemenea, se va ține cont și de gradele de gătire ale preparatului în concordanță cu gradele maxime existente per fiecare cuptor în parte.

2p. Să se testeze soluția prin crearea unui număr de 5 comenzi și să se asigneze optimizat comenzile la cuptoarele existente. Să se apeleze metoda de afișare a tuturor comenzilor aflate în așteptare pentru fiecare cuptor existent în restaurant.