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 bancomat.

5p. În cadrul unei bancomat, se doreste implementarea modulului de retragere numerar. Clientul are posibilitate, dupa ce introduce cardul alaturi de parola sa, sa ceara retragerea unei sume de bani. Bancomatul poate detine doar bancnote de 50, 20 si 10 lei. Daca cererea nu poate fi realizata (nu exista suficiente bancnote in bancomat, suma nu poate fi formata prin compunere de bancnote existente) atunci se va arunca o exceptie si nu se va retrage nicio bancnota din bancomat. Implementarea trebuie sa permita adaugarea de noi tipuri de bancnote cu minim de modificare de cod. De asemenea, trebuie sa existe posibilitatea sa a putea schimba ordinea de preluare bancnote in functie de valoarea lor.

3p. Pattern-ul este testat în main() prin popularea bancomatului cu un numar de bancnote din fiecare tip. Ulterior, se va incerca retragerea de numerar; in unul din cazuri, extragerea sa fie realizata, iar, in celalalt, sa includa un context in care sa nu se poata retrage suma respectiva.

1p. Sa se motiveze in scris alegerea design pattern-ului care modeleaza cel mai bine situatia enuntata. Motivarea trebuie sa contina elementele cheie care denota utilizarea respectivului design pattern.

5p. Clientul are posibilitatea de a alege tipul de bancnote pe care le doreste sau poate opta pentru utilizarea tuturor tipurilor de bancnote existente in bancomat. Daca clientul selecteaza o alegere personalizata, aceasta se citeste de la tastatura. Se va utiliza interfata IClient.

3p. Pattern-ul este testat în main() prin utilizarea unui client si setarea la nivelul sau a celor doua optiuni de parametrizare a unei retrageri.

1p. Sa se motiveze in scris alegerea design pattern-ului care modeleaza cel mai bine situatia enuntata. Motivarea trebuie sa contina elementele cheie care denota utilizarea respectivului design pattern.