CTS2021 - S3 - Robot software

Calitate si testare Software (Academia de Studii Economice din Bucure□ti)

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 corect 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 legătură 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.pattern.X*, *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 aiba legatura 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 robot software care să digitalizeze serviciile de secretariat ale unei facultăți.

2p. Robotul va prelua cererile studenților, va răspunde la solicitări de informații și se va interconecta cu alte servicii software din facultate. Pentru a eficientiza integrarea robotului cu alte soluții se va dezvolta un API care să permită dezvoltatorilor să interacționeze cu robotul software prin intermediul unui obiect unic ce expune interfața ***IRobotSoftware***. Implementați clasa care va permite acest lucru, având în vedere că în sistem va exista doar o singură instanță a robotului software.

2p. Să se testeze soluția, prin implementarea interfeței primită și prin demonstrarea faptului că încercarea de a construi mai multe instanțe de tip robot software nu va conduce la existența mai multor obiecte.

5p. Cererile gestionate de robotul software parcurg un flux intern și vor afișa un mesaj de status în funcție de starea în care se găsesc. Starea unei cereri este influențată de parcurgerea unor etape descrise de interfața ***ICerereStudent***. Starea cererii este modificată doar intern prin parcurgerea unei etape, nefiind permisă modificarea ei direct. De exemplu, mesajul “Cerere primită” va fi afișat doar după ce este confirmată (acțiunea *confirmare()*). Inițial cererea este în starea “Cerere trimisă pe flux” Mesajele aferente unui status se pot modifica în timp, diferă de la etapă la alta sau depind de particularitățile unei facultăți. O cerere care depășește un termen de 10 zile trece automat în starea “Respinsă”. În etapa de verificare a cererii, cererile fără motiv vor fi “Respinse” iar cele ce sunt ok sunt trimise spre “Avizare”. Găsiți o soluție care să permită gestiunea mesajelor afișate pentru fiecare stare, independent de clasa ce gestionează cererea studentului.

2p. Pattern-ul este testat în main() prin definirea și utilizarea unei cereri care trece prin etape, afișând mesajul specific stării de la acel moment. Testați și situațiile aferente unei cereri care a depășit termenul de 10 zile, a unei cereri complete și a unei cereri fără motiv.

7p. Pentru a gestiona situațiile în care numărul mare de solicitări nu poate fi procesat în timp real, robotul software implementează un mecanism care să îi permită procesarea lor ulterioară, asincronă, în funcție de tipul lor. Se consideră că solicitările pot fi Urgente sau Normale (***TipCerere***). Solicitările (descrise de

tip și denumire) sunt procesate de diferite module ale robotului software, module ce implementează interfața ***IProcesatorCerere***. Se iau în considerare cel puțin 2 categorii de cereri (adeverință, eliberare diploma, plata taxa, etc) ce vor fi procesate de modulele aferente. Să se găsească o soluție eficientă care să permită procesarea cererilor primite în sistem. Cererile de tip Urgenta vor fi procesate cu prioritate. Dacă în sistem nu mai există cereri urgente atunci vor fi procesate și cele normale.

2p. Să se testeze soluția prin definirea a cel puțin 3 solicitări (de tipuri și categorii diferite) care să fie înregistrate în sistem. Să se demonstreze că cerere urgenta este procesata cu prioritate, chiar daca nu a fost prima înregistrată în sistem.