Programare orientată pe obiecte – Test de laborator **02/06/2025**
Timp de lucru: 90 minute **Oficiu: 1p**
Seriile 13,14,15

Enunț

Întrucât aplicațiile anterioare au avut un succes extraordinar, organizația Q&V dorește să își extindă domeniul de activitate deschizând un restaurant. Pentru a eficientiza operațiunile, aceștia doresc să implementeze o aplicație.

Aplicația va permite adăugarea comenzilor clienților. Fiecare comandă are un identificator unic și un număr dinamic de produse. Fiecare produs are un nume și un gramaj. Fiecare produs poate fi fie o băutură (memorându-se dacă este la sticlă sau nu), desert (menționându-se formatul servirii, felie, cupă sau porție) sau burger (listând o serie de ingrediente, de exemplu roșii, varză, etc.). Produsele sunt predefinite și se găsesc în cadrul unui meniu. Pentru fiecare dintre ele se poate calcula valoarea de energie necesară pentru prepararea cu formula:

- Băutură -> dacă este la PET, 25, dacă nu 0.25 * gramaj
- Desert -> dacă este felie 25, porție 0.5 * gramaj sau cupă 2 * gramaj
- Burger -> gramaj * 0.25 * numărul de ingrediente

Pentru a demonstra funcționalitatea aplicației va exista un simulator. Acesta este construit pe baza unui ciclu ce include ca toți angajații să participe în sarcinile magazinului. Acestea sunt: preluarea, prepararea și livrarea comenzilor. Întrucât Q&V dorește să aplice tehnici de management modern, inspirate din IT, aceștia vor să aibă doar echipe cross funcționale, astfel toți angajații pot să facă toate acțiunile, însă aceștia au eficiență diferită.

Un angajat are un număr de puncte de energie, inițial 100. Aceștia consumă energie pentru a face o anumită acțiune. După un ciclu, se întorc înapoi la 100 energie.

La începutul unui ciclu se introduc comenzile. Pentru a lua o comandă, un angajat consumă 100 de puncte de energie. Un angajat poate sa preia maxim o comandă.

După etapa de preluare a comenzilor, sunt preparate comenzile. Angajații își consumă energia pentru a prepara comenzi. (De exemplu, un angajat cu 100 energie va scădea 100 din energia necesara preparării comenzii).

Dacă o comandă este terminată, atunci ea poate fi înapoiată către client la un cost de 100 energie.

Există mai multe tipuri de angajați, fiecare având anumite perk-uri:

- Casieri => cost de energie doar de 25 pentru a prelua o comandă
- Livratori => cost de energie doar de 25 pentru a livra o comandă
- Bucătari => dublează cantitatea de energie pe care pot să o producă asupra unui produs, iar aceștia primesc 100 energie la finalul ciclului în loc să întoarcă la 100 (aka pot depăși acel 100 energie inițial).

Cerințe:

- Afișarea numărului de angajați pentru fiecare tip în parte
- Implementarea simulării și afișarea unui ciclu (sub orice fel).
- Crearea unei opțiuni ca între cicluri să poți selecta comenzi care sunt prioritare, și ar trebui să fie terminate primele
- Optimizarea ciclului (se va prioritiza ca fiecare angajat sa își facă rolul, vezi exemplu pentru varianta neoptimizată)
- Posibilitatea de a selecta strategia de alegere a comenzilor care sunt preparate (de exemplu FIFO, comanda cea mai "grea" prima, comanda cea mai ușoară prima)

Atenție! Precizări pe verso

Programare orientată pe obiecte – Test de laborator **02/06/2025**
Timp de lucru: 90 minute **Oficiu: 1p**
Seriile 13,14,15

Exemplu orientativ:

```
angajati = [Livrator(100), Bucătar(125), Bucătar(100), Casier(100)]
                        // un bucătar are 125 întrucât a rămas cu 25 energie la ciclul
                precedent
comenzi = []
== START ==
Adăugare comanda? (1/0) 1
comenzi = [Comanda(650 energie)] // Livrator citește comanda devine Livrator(0)
Adăugare comanda? (1/0) 0 // la primul refuz, se vor evita adăugări ulterioare
comenzi = [Comanda(400)] // Bucatar consuma energie devine Bucatar(0)
comenzi = [Comanda(200)] // Bucatar consuma energie devine Bucatar(0)
comenzi = [Comanda(100)] // Casier consuma energie devine Casier(0)
== END ==
angajati = [Livrator(100), Bucătar(100), Bucătar(100), Casier(100)]
== START ==
Adăugare comanda? (1/0) 0
comenzi = [Comanda(0)] // Livrator consuma energie devine Livrator(0)
comenzi = [] // Bucatar livreaza comanda, devine Bucatar(0)
== END ==
angajati = [Livrator(100), Bucătar(100), Bucătar(200), Casier(100)]
```

Precizări:

0. Sursa predată trebuie să compileze. Erorile mici se pot trece cu vederea. Înainte de predarea surselor, studenții vor pune în comentarii eventualele părți din program care au erori de compilare sau nu funcționează corespunzător. Codul comentat se poate puncta parțial dacă este corect.
1. Sursa predată trebuie să respecte condițiile expuse în **Regulamentul de colocviu** care a fost trimis către studenți (via Teams) în data de 01/06/2025.
2. Se acceptă și soluții parțiale, care nu respectă toate cerințele din enunț, dar sunt funcționale. Acestea vor fi punctate corespunzător.
3. În implementarea programului se vor utiliza cât mai multe dintre noțiunile de programare orientată pe obiecte, care au fost studiate pe parcursul semestrului și care se potrivesc cerințelor din enunț.
4. Orice tentativă de fraudă se va pedepsi conform regulamentelor Universității.
   Se interzice utilizarea sau rularea în fundal a aplicațiilor care pot fi folosite pentru a comunica. În cazul observării acestora, nota la examen va fi 1.
5. Daca rezolvarea nu este făcută cu clase sau daca datele din clase sunt publice, nota va fi 1.
6. Funcționalitățile aplicației vor fi implementate la alegere sub formă de meniu interactiv sau ca apeluri în funcția main pentru fiecare cerință.