1. Cerinte pentru etapa a 2-a

1. **Sa se puna accent pe concepte ale mostenirii:**
   1. **Daca aveti entitati (destinatii, haine, etc) cu date comune, mosteniti o supra clasa comuna (spre exemplu sunt tipuri de spectatori sau destinatii, dar ei au in comun numele si la destinatii au in comun un ranking sa spunem)**
   2. **Daca mai multe obiecte au comportament similar si functii similare (produse masurate la bucata cu stoc bucata, folosim clase care contin functii utile**
   3. **Avem nevoie de polimorfism dinamic (clase care au o supraclasa comuna, cu functie virtuala) pentru a simplifica majoritatea problemelor. Asta ne permite sa punem mai multe clase similare intr-o singura lista**
2. **Primiti bonus daca folositi std::string si std::vector.**
3. **Prima parte a punctajului, de 50%, este modelarea claselor -> adica sa aveti clase pentru toate datele din cerinta.**
4. **A doua parte a punctajului, tot de 50%, il castigati prin a implementa cerintele care tin de functionalitati.**
   1. **E.g**  “Clientul soseste cu o lista de cumparaturi cu bunuri. Magazionerul ii comunica clientului pretul total iar acesta decide daca cumpara sau nu. Daca cumpara se realizeaza, se produce schimbul bunuri contra bani. La finalul zilei se realizeaza inchiderea zilei, afisand totaluri de bunuri vandute si banii obtinuti.”
5. **Castigati bonus de pana la 25% pentru codestyle ->**
   1. **Sa folositi fisiere .h si .cpp**
   2. **naming conventions:**
      1. **clase denumite cu *PascalCase***
      2. **nume de variabile si functii folosind *camelCase***
      3. **denumirea functiilor sa se faca folosind verbe, spre exemplu getTicket, reserveRoom etc**
      4. **Sa minimizati folosirea this->, prin a alege una din doua variante: prefixati orice date member cu m\_, sau denumiti parametrii ca \_numeParametru. In C++ ambele sunt folosite, dar m\_ poate fi mai sugestiv, deoarece este folosit pentru variabile membre iar la cele statice nu este folosit, ceea ce clarifica usor ce tip de variabile folosesti intr-o metoda a clasei**
6. **ATENTIE la lucrurile care sunt cerute la Colocviu:**
   1. **In final creezi o arhiva NUME\_PRENUME.rar sau NUME\_PRENUME.zip**
      1. **Care NU :)) contine executabilul nici fisierele generate de CLion pt debugging (e un folder)**
      2. **Contine toate fisierele .h si .cpp din proiect. In cazul examenului nu sunt cerute fisierele CMakeLists.txt din simplul motiv ca nu toti studentii lucreaza cu CLion.**
   2. **Fara variabile globale**
   3. **Toate datele membre sunt private (cele statice - nu neaparat. Ganditi-va ca o utilizare a variabilelor statice este inclusiv de a scrie constante precum qt::yellow  si qt:red (culor) in unele frameworkuri)**

Curierat rapid:

O firma de curierat are mai multe vehicule in teritoriu (scutere, masini, dube). Fiecare are o capacitate (o masa maxima admisa si un volum total posibil). Unele dube sunt frigorifice si pot transporta bunuri reci. Fiecare vehicul se afla intr-un anumit punct (x,y).

Firma (dispeceratul) primeste comenzi de livrare a unor colete (mai multe articole cu masa / volum diferite, unele dintre ele putand fi reci) de la un client (aflat la alte coordonate x,y) catre o destinatie (din nou x,y). Intreg coletul este preluat de un singur livrator.

Soferii se deplaseaza cu viteze diferite (scutere – 20 Km/s, masini – 10 Km/s, dube – 5 Km/s) catre destinatie, in linie dreapta (scutere) sau manhattan (masini, dube). Odata livrat un colet, vehiculul ramane la destinatie daca nu mai are alte colete de livrat. Coletele trebuie sa ajunga „la timp”. Pentru diferite colete la timp poate insemna: pana la o anumita secunda din zi, intr-un interval de timp maxim (de la preluarea comenzii/preluarea coletului), cat mai repede (fara limita de timp).

Firma poate aplica 3 strategii de alocare a coletelor (e.g. vehiculul cel mai apropiat de client, vehiculul care – conform strategiei sale curente – poate livra cel mai repede coletul, vehiculul cel mai putin incarcat). Vehiculele se aleg dintre cele capabile sa transporte coletul (au spatiu) si ar reusi conform strategiei curente sa il livreze la timp. Daca nu exista nici un astfel de vehicul, clientul este refuzat. De asemenea, soferul vehiculului poate opta pentru 3 strategii de livrare/ridicare (coletul cel mai urgent primul, coletul cel mai apropiat de pozitia curenta, first-come-first-served). Sa se simuleze operatiunile firmei de curierat intr-o zi.

Precizari:

* Un vehicul nu isi schimba destinatia pe parcursul unui drum (doar in momentele in care ridica/livreaza un colet).
* Strategia soferului poate trata unitar (la fel) ambele tipuri de operatiuni (ridicare si livrare) sau poate sa faca distinctie intre ele (e.g. mai intai livrari apoi ridicari).
* Strategiile de alocare pot fi schimbate de manager pe parcursul unei zile. La fel si strategiile fiecarui sofer (tot de manager) – insa fara ca un colet sa ajunga sa nu mai poata fi livrat la timp.
* Performanta algoritmului de alocare nu este cruciala pentru aceasta cerinta la acest curs.