

Enunț

O un producător de masini dorește să pornească o afacere și în acest sens el dorește să creeze un sistem de gestionare a modelelor de autoturisme pe care le fabrică. Producătorul dorește să realizeze aplicația în C++ folosind concepte de OOP.

Producătorul își propune să creeze mașini care funcționează pe bază de combustibili fosili (care pot fi de două tipuri: benzină și motorină), mașini electrice și mașini hibride (care pot fi alimentate atât electric, cât și benzină)

Despre un model de mașină, producătorul dorește să rețină următoarele informații: anul începerii producției, numele modelul, viteza maximă și greutatea.

În plus, pentru un model de mașină pe bază de combustibili fosili se mai rețin: tipul combustibilului (benzină sau motorină) și capacitatea rezervorului. Pe de altă parte, pentru un model electric se reține adițional: capacitatea bateriei.

Autonomia unei mașini dintr-un model se calculează:

- pentru modele pe combustibili fosili: raportul dintre capacitatea rezervorului și radical de ordinul doi din greutate
- pentru modele electrice: raportul dintre capacitatea bateriei și greutate la pătrat
- pentru modele hibride: suma celor două definite mai sus

Producătorul dorește de asemenea de la program să memoreze o listă a tranzacțiilor efectuate de-a lungul timpului cu diverși clienți. O tranzacție este definită prin: numele clientului, data tranzacției și o listă de modele achiziționate.

Un exemplu de tranzacție: {Popescu, 5 5 2019, [Masina_1, Masina_2, Masina_3]}, unde Masina_i sunt obiecte/pointeri/referințe din clase reprezentând modele, **NU** nume de modele.

Cerințe:

- I. Să se implementeze clasele necesare astfel încât aplicația să funcționeze.
- II. Programul este prevăzut cu un meniu interactiv în care administratorul are posibilitatea de a alege dintre următoarele opțiuni:
 1. Adaugă un nou model mașină citind de la tastatură tipul mașinii și apoi câmpurile specifice ei
 2. Adaugă o tranzacție
 3. Afișează cel mai vândut model
 4. Afișează modelul cu autonomia cea mai mare
 5. Aduce o optimizare unui anumit model crescându-i viteza maximă cu un anumit procent.

Precizări

1. Timpul de lucru este de 90 de minute.
2. La sfârșitul timpului de lucru, studenții vor trimite pe adresa examen.oop.fmi@gmail.com fișierul sursă cu extensia cpp. Acesta trebuie să conțină pe primul rând un comentariu cu numele și prenumele studentului, grupa și compilatorul folosit.
3. Sursa predată trebuie să compileze. Sursele care au erori de compilare nu vor fi luate în considerare. Înainte de predarea surselor, studenții vor pune în comentariu eventualele părți din program care au erori de compilare sau nu funcționează corespunzător.
4. Se acceptă și soluții parțiale, care nu respectă toate cerințele din enunț, dar sunt funcționale. Acestea vor fi depunctate corespunzător.
5. 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.

6. Condițiile minimale de promovare a testului sunt ca programul să fie scris cu clase și să citească modelele de mașini unul câte unul, să le memoreze datele și apoi să le afișeze.
7. Orice tentativă de fraudă se va pedepsi conform regulamentelor Universității.

Barem

1. Se acordă 1 punct din oficiu.
 2. Dacă sursa nu compilează, nu se mai acordă niciun alt punct (nota 1).
 3. Se acordă 4 puncte pentru respectarea condițiilor minimale, prevăzute în enunț, (programul să fie scris cu clase și să ruleze corect funcția main() în varianta demo) astfel, dintre care:
 - 2,5 puncte pentru definirea claselor:
 - a. Clasa Model
 - constructor de inițializare fără parametri - 3 x 0,1 p
 - constructor de inițializare parametrizat - 3 x 0,1 p
 - destructor virtual - 3 x 0,1 p
 - definirea corectă a câmpurilor clasei - 3 x 0,1 p
 - b. implementarea metodei calculAutonomie() în clasele ModelFosil, ModelElectric, ModelHibrid - 3 x 0,1 p
 - c. Clasa Tranzactie
 - constructor de inițializare parametrizat - 0,2 p
 - constructor de copiere - 0,2 p
 - destructor - 0,2 p
 - operator de atribuire - 0,2 p
 - implementarea listei de Model* 0,2 p
 - iar 1.5 puncte se acordă pentru respectarea condițiilor minimale, prevăzute în enunț, astfel:
 - d. 0.5 puncte dacă modelele de orice tip (combustibili fosili, electrice și hibride) sunt citite corect, cu toate informațiile aferente
 - e. 0.5 puncte dacă modelele de orice tip (combustibili fosili, electrice și hibride) sunt memorate corect într-o listă sau într-o altă structură
 - f. 0.5 puncte dacă modelele de orice tip (combustibili fosili, electrice și hibride) sunt afișate corect, cu toate informațiile aferente.
- Pentru oricare dintre cerințele a., b., și c., dacă cerința este îndeplinită parțial, se acordă 0,25 puncte.
4. Se acordă 1 punct pentru folosirea corectă în contextul dat a supraîncărcării operatorilor. Dacă cerința este îndeplinită parțial (sau doar pentru << si >>), se acordă 0,5 puncte.
 5. Se acordă 1 punct pentru definirea unei ierarhii de clase (**Model** -> **ModelFosil**, **ModelElectric**, **ModelHibrid**). Dacă cerința este îndeplinită parțial, se acordă 0,5 puncte.
 6. Se acordă 1 punct pentru definirea corectă a unei ierarhii de clase de tip romb (moștenire multiplă și virtuală). Nu se acordă punctaje parțiale.
 7. Se acordă câte 0,4 puncte pentru rezolvarea corectă a cerințelor II.1., II.2., II.3., II.4., II.5., II.6. din enunț. Nu se acordă punctaje parțiale.
 8. Se scade 1 punct pentru utilizarea incorectă a constructorilor și destructorilor. Dacă cerința este încălcată parțial, se scad 0,5 puncte.
 9. Se scade 1 punct pentru utilizarea incorectă a încapsulării (câmpurile private și metode publice). Dacă cerința este încălcată parțial, se scad 0,5 puncte.
 10. Se acordă 1 punct pentru folosirea corectă în contextul dat a cel puțin 2 elemente deosebite de OOP (clase șablon, tratarea excepțiilor, design patterns, etc.). Dacă cerința este îndeplinită parțial, se acordă 0,5 puncte.
 11. Se acordă 0,5 puncte pentru folosirea corectă în contextul dat a metodelor virtuale. Nu se acordă punctaje parțiale.

12. Se acordă 0,5 puncte pentru folosirea corectă în contextul dat a unei clase abstracte.

Observații de evaluare

1. Nota maximă este 12.
2. Erorile minore, care sunt cauzate de neatenție și care nu se repetă în mai multe locuri (de ex., a uitat să scrie un caracter și nu îi compilează, a greșit un semn sau un coeficient într-o formulă, nu afișează primul sau ultimul element dintr-o listă etc) pot fi corectate și trecute cu vederea (se poate acorda punctajul maxim prevăzut pentru o cerință îndeplinită corect din cauza unei erori minore).