Cerințe proiect laborator POO

Sumar

- notă bazată pe un proiect individual cu trei etape de evaluare
- note de la 1 la 12 pentru fiecare etapă (10 de bază + maxim 2 puncte bonus)
- nota finală este media notelor la fiecare etapă și trebuie sa fie minim 5
- tema aleasă trebuie confirmată cu mine pe email / teams sau la laborator și trecută în acest tabel **până la sfârșitul săptămânii 5 (26.03.2023)**
- proiectele vor fi corectate prin prezentarea la laborator (pentru cazuri speciale, adică studenți care nu pot ajunge la laborator, îmi puteți scrie pe mail)
- baremele de notare pentru fiecare etapă se găsesc mai jos în document
- **bonus** - +0.5p la nota finală pentru cine rezolvă corect și frumos scris toate exercițiile din laboratoare

Date finale pentru fiecare etapă

Codul pentru fiecare etapă se poate scrie până la:

- sfârșitul săptămânii 6 (02.04.2023) pentru etapa 1
- sfârșitul săptămânii 9 (30.04.2023) pentru etapa 2
- sfârșitul săptămânii 12 (21.05.2023) pentru etapa 3

Prezentările de proiect se pot susține oricând până în săptămâna 13.

Prezentare generală

Nota voastră la laboratorul de POO se acordă pe baza unui **proiect individual**, dezvoltat pe parcursul primelor 13 săptămâni din semestru[1]. Proiectul va fi evaluat în **trei etape**, cu cerințe și deadline-uri distincte pentru fiecare[2].

Proiectul constă în dezvoltarea unei **aplicații C++** care **să gestioneze un model de date**, folosind paradigma de **programare orientată pe obiecte**. Va trebui să definiți atât clasele care să reprezinte entitățile din modelul vostru, cât și metodele/funcțiile corespunzătoare care să implementeze „logica de business”.

[1] În ultima săptămână veți da un colocviu (test pe calculator), până la acel moment trebuie să aveți situația la laborator încheiată.

[2] Cerințele sunt gândite în așa fel încât să continuați programul din etapa anterioară, adăugând funcționalități în plus la codul existent.

Notare

Pentru fiecare etapă, veți primi **o notă de la 1 la 12**. Nota 10 se acordă pentru implementarea corectă a tuturor **cerințelor de bază**, putând primi până la două puncte în plus dacă rezolvați și **cerințele suplimentare** (vor fi indicate corespunzător).

Nota finală este media notelor de pe fiecare etapă. Trebuie să aveți minim **nota 5** ca să promovați laboratorul și să puteți participa la colocviu/examen.

Alegerea temei

Recomandarea mea este să alegeți o temă care vă pasionează, ca să fiți motivați să lucrați la proiect și să îl duceți până la capăt.

Exemple de teme

- Magazin online (produse, cumpărători, comenzi, reduceri etc.)
- Gestiunea resurselor umane (firmă, angajați, echipe, salarii etc.)
- Gestiunea unei clinici/unui spital (medici, pacienți, consultații, medicamente etc.)
- Project management (proiect, task, echipă, membru al echipei etc.)
- Gestiunea școlarității (studenți, note, discipline, profesori etc)
- Bibliotecă (carte, autor, categorie, împrumut etc.)
- Music player (album, artist, melodie, playlist etc.)
- Grădină zoologică (animale, bilete, hrană pentru animale etc.)
- Formula 1 (echipe, mașini, piloți, raliuri etc.)
- Cinematograf (filme, actori, vizionări, bilete etc.)

Puteți alege orice alt domeniu doriți. Indiferent de tema aleasă, va **trebui să o confirmați** cu mine (în persoană când veniți la laborator, sau pe e-mail/Teams dacă nu puteți ajunge) **până la sfârșitul săptămânii 6** de facultate.

De preferat ar fi să vă alegeți teme distincte. Cel mult doi studenți pot avea o temă identică/similară. Chiar și în acest caz, vor trebui să aibă o structură diferită a claselor.

Important: este permis și chiar recomandat să vă ajutați între voi la proiecte, însă nu este tolerat să copiați cod unii de la alții sau din proiecte publice de pe internet.

Nu uitați că atât colocviul cât și examenul sunt individuale, deci va trebui să fiți capabili să le rezolvați singuri.

Recomandare: după ce v-ați ales tema și ați confirmat-o, puteți începe prin a vă face o „schiță" cu clasele de care veți avea nevoie, ce date și metode vor reține fiecare și care vor fi legăturile dintre ele. Puteți face asta pe hârtie sau folosind o aplicație cum ar fi Excalidraw.

Recomandare: familiarizați-vă cât mai curând cu și începeți să folosiți debuggerul din mediul vostru de lucru. O să vă petreceți 20% din timp scriind efectiv cod și peste 80% din timp încercând să înțelegeți de ce nu funcționează cum v-ați fi așteptat.

Cerințe etapa 1

Deadline: cerințele trebuie implementate până la sfârșitul **săptămânii 6** de facultate (**02.04.2023**). Pentru a fi punctați, va trebui să le și prezentați la laboratorul din **săptămâna 6**.

Criterii generale **(1p)**

- Trebuie să aveți în `main` un cod corespunzător care să testeze/utilizeze funcționalitățile implementate.
- Clasele/metodele care nu sunt utilizate sau la care nu se face referire nicăieri **nu vor fi punctate**.
- Pentru etapa 1, **nu** aveți voie să folosiți clasele container din STL (`std::vector`, `std::string` șamd). Puteți refolosi în schimb clasele implementate de voi la laborator (este recomandat să le refolosiți).
- Încercați să păstrați codul curat:
  - Folosiți nume de variabile/funcții/tipuri de date cu sens (e.g. `nr_angajati` în loc de `n`).
  - Folosiți formatarea automată oferită de editorul vostru de text și încercați să păstrați un stil uniform.
  - Folosiți mai degrabă mai multe clase/metode mai scurte, decât o singură clasă/metodă foarte lungă.

Versionarea codului **(1p)**

- Codul sursă **trebuie** să fie încărcat pe GitHub.

- Trebuie să îmi dați și mie **acces de citire** la repository-ul în care aveți proiectul, dacă nu este public.

  Instrucțiunile pentru cum puteți adăuga colaboratori la un repo se găsesc aici. Mă puteți adăuga prin username (`@JustBeYou`) sau prin e-mail (`mihailferaru2000@gmail.com`).

- **Recomandare:** adăugați de la început un fișier `.gitignore` la repo-ul vostru ca să nu includeți accidental fișierele compilate / binare în Git. Puteți găsi aici un exemplu de fișier `.gitignore` pentru C++.

- **Recomandare:** păstrați commit-urile concise și independente. Găsiți aici un set de bune practici pentru commit-urile de Git.

Documentație **(1p)**

- În repo-ul de pe GitHub trebuie să aveți și un fișier `README`, de preferat formatat cu Markdown, în care să includeți cel puțin:
  - **Numele** proiectului
  - **Tema** aleasă
  - O listă cu **clasele** pe care le-ați implementat și o scurtă descriere a ce reprezintă fiecare
  - O listă cu **funcționalitățile** pe care le are aplicația voastră la momentul actual (ex.: „capabilă să citească și să rețină o listă de angajați”, „poate calcula prețul mediu al produselor aflate în stoc” etc.)

- **Recomandare:** actualizați acest fișier pe parcurs ce dezvoltați proiectul. Vă va fi mult mai ușor să-l prezentați altor persoane.

Clase **(2p)**

- Trebuie să respectați **principiul encapsulării** (nu aveți voie să aveți date membre publice). Metodele pot fi publice sau private.

- Trebuie să definiți **minim 5 clase**, relevante pentru tema aleasă.

  Sunt luate în considerare și clasele utilitare, cum ar fi o clasă pentru șiruri de caractere sau una pentru vectori alocați dinamic.

- **Toate clasele** vor avea constructori cu parametri sau fără care să inițializeze membrii clasei.

- **Minim 3 dintre clase** trebuie să fie corelate prin **compunere** (ex. să aveți o dată membru de tip `Adresă` în clasa `Contact`, să aveți un vector de `Angajat` în clasa `Companie` etc.)

Metode **(2p)**

- **Minim 3 metode** care operează cu datele membru ale claselor (e.g calculează prețul unui produs aplicând o reducere, returnează salariul mediu al angajaților din firmă, actualizează TVA-ul unei facturi și recalculează totalul etc.). Nu se vor lua în considerare getter-ii și setter-ii simpli și fără logică "de business" [3].
- **Minim o metodă** va fi supraîncărcată (e.g. reducerea aplicată poate fi un întreg de la 0 la 100 sau un număr real între 0 și 1).

Citire și afișare **(1p)**

- Pentru **minim o clasă** trebuie să implementați o metodă de **afișare** și una de **citire** supraîncărcând operatorii `operator<<` și `operator>>`.

Alocare dinamică **(1p)**

- **Minim o clasă** trebuie să facă alocare dinamică de memorie (fie pentru un singur obiect, fie pentru un vector de obiecte).
- **Minim o clasă** care face alocare dinamică va avea implementat un constructor de copiere și operatorul `operator=`.
- Toată memoria alocată dinamic trebuie **eliberată** corespunzător. Veți fi depunctați pentru *memory leaks*.
- **Minim o clasă** trebuie să aibă un **destructor** (netrivial, adică cu eliberare de memorie).

Bonus

Fiecare cerință bonus valorează **1 punct**.

- Implementați un **meniu interactiv** în `main`, care să permită:
  - **Citirea** de la tastatură și **crearea** a **cel puțin 3 tipuri de obiecte** dintre cele definite de voi;

[3] Ce este business logic-ul găsiți aici: https://en.wikipedia.org/wiki/Business_logic.

  - **Afișarea** acestor obiecte, după ce au fost citite de aplicație;
  - **Apelarea unei metode** pe ele (alta în afară de cea de afișare).
- Implementați **constructorul de mutare** și supraîncărcați `operator=` **de mutare** pentru cel puțin o clasă care gestionează memorie alocată dinamic.

Oficiu (1p)