# Proiect 2

Anul Universitar 2023 – 2024, Semestrul II

Programare Orientată pe Obiecte

GRUPA 133

**Cuprins**



**Termen limită predare: Înainte de vacanța de 1 mai (puteți și în vacanță, dar nu după).**


Autor: Daniel Wagner

## I. Important înainte să parcurgeți materialul

Citire completa: https://github.com/mcmarius/poo/blob/master/tema-2/README.md

tl;dr https://github.com/mcmarius/poo/blob/master/tema-2/README.md#exemplu-complet-func%C8%9Bii-virtuale

și https://github.com/mcmarius/poo/blob/master/tema-2/README.md#exemplu-complet-excep%C8%9Bii

## II. Resurse necesare:

- Laboratoarele 1-8
- TODO: de adăugat link-uri oficiale, tutoriale, etc.

**Observație**: Cerințele generale obligatorii sunt cele pentru nota 10 maximă (sau mai puțin), în funcție de cât de mult ați implementat din lista de cerințe. Pentru obținerea unei note până la 12, trebuie să îndepliniți criteriile de bonus ilustrate.

## III. Cerințe speciale pentru obținerea notelor maxime (12):

- Codul trebuie să fie bine organizat, cu împărțirea corectă în fișiere **.h** și **.cpp**, care se află în **include/** și **src/**.
- Utilizarea eficientă a containerelor STL, a mecanismului de excepții.
- Utilizarea corectă a unei interfețe, a unei clase abstracte, si a obiectelor de tip derivat din clasa abstractă, cu toate cerințele generale implementate
- Documentația codului și comentariile relevante pentru metodele și clasele importante.
- Diagrama cu ierharia de clase rezultată din proiectul vostru
- TODO: alte cerințe speciale pentru notă maximă?

# IV. Cerințe generale obligatorii

- O interfață care definește funcțiile virtuale pure
- O clasa abstractă (care implementeaza anumite metode ale interfeței, dar nu pe toate)
- Variabila și funcție statică
- Variabila constantă
- 2-3 clase care moștenesc clasa abstracta, vector/listă/colecție de pointeri către bază
- Utilizarea corectă a claselor polimorfice, de ilustrat în main() pe o colecție de obiecte de tip Baza cu dynamic_cast, să se apeleze toate metodele obiectului de tip derivată.
- Resurse alocate dinamic în cel puțin una din clase + regula celor 5 (cc/op=/destructor/constructor mutare/op= pentru mutare)
- Destructor virtual pentru resursele alocate dinamic, obligatoriu cel puțin o clasă cu resurse alocate dinamic (și eliberate corespunzător).
- Vector/List/Colectie de pointeri către Bază cu downcasting cu dynamic_cast
- Operator de afișare definit în clasa abstractă care apelează o funcție virtuală de afișare, a obiectului derivat, precum în lab06.
- Minim 2 funcții virtuale diferite de Destructor și Afisare()
- Cel puțin o funcție virtuala care este suprascrisă doar în clasa abstractă și în niciuna din derivate
- 1-2 funcții comune non-virtuale definite in clasa abstractă
- Minim o clasă proprie pentru excepții, și folosirea mecanismului de excepții pe parcursul programului.
- Toate atributele vor fi definite **private** (sau **protected** în cazul moștenirii). Important este să nu fie niciodată declarate atribute **public** în cadrul claselor. Însă, pentru structuri, pot ramâne publice pentru accesul mai ușor, spre exemplu:

  **struct Punct2D { int x; int y; }**

  Nu are sens să facem getters & setters și le declarăm publice (implicit în structuri).
- Utilizarea static_cast unde este nevoie
- Metodele interne clasei (funcțiile ajutătoare), de asemenea **private/protected** (depinde dacă avem nevoie de ele în clasa derivată) – nu are sens să le poată apela utilizatorul, ci doar voi din interior.

- Utilizare **const** peste tot unde este posibil (în mod special când trimiteți prin referință fără să modificați obiectul sau la metodele care nu modifică obiectul **this**)
- Toate metodele definite trebuiesc și apelate/testate în cadrul claselor sau în **main()**, altfel nu are sens să le fi definit.
- Fără variabile globale și **fără cale absoluta (C:/...), doar cale relativa (../../dir)**
- **FĂRĂ using namespace std** – niciodata în **.h**, tolerabil în **.cpp**, dar tot neindicat; **Observație**: sunteți liberi să îl folosiți la colocviu, unde orice secundă câștigată conteaza.
- Proiectul să fie încărcat **pe GitHub fără erori de compilare**, cu cât mai multe commit-uri relevante pe parcurs cu mesaje descriptive, care să ilustreze progresul. Munca pe parcurs va fi luată în considerare subiectiv la notare în plus. Insă, studenții cu proiecte cu un singur commit sau doar pe final vor fi întrebați mai în amănunt la prezentare.
- README.md care explică proiectul (numele proiectului, scopul proiectului, structura claselor, funcționalități)
- .gitignore pentru fișierele pe care nu doriți să le includeți (nu includeți alte fișiere decât cele necesare proiectului, și **să nu includeți fișiere executabile pe GitHub sau fișiere obiect ".o"**, doar cod sursă!!)
- Fișier de input pentru datele citite de la tastatură. Fișierul de input este necesar încât să nu fiu nevoit să introduc manual datele pentru a vă testa proiectul. Puteți să vă redirecționați și voi input-ul din fișier în terminal din același motiv (exemplificat în Lab04).

**Opțional pentru bonus**:

Fișier Gr133_*Nume_Prenume*_Proiect2.txt trimis în particular pe Teams cu ce nu ați reușit să implementați și unde v-ați blocat, rezumat general scurt al conceptelor învățate din lucrul în cadrul proiectului, și ce nu s-a înțeles din cerințe, plus ce mai considerați voi relevant. Util pentru feedback dar și autoevaluare.