

## **Proiect 2**

Anul Universitar 2023 – 2024, Semestrul II

Programare Orientată pe Obiecte

GRUPA 133

### **Cuprins**

- I. Important înainte să parcurgeți materialul**
- II. Resurse necesare**
- III. Cerințe speciale pentru obținerea notelor maxime (12)**
- IV. Cerințe generale obligatorii**

**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 (informații suplimentare – posibil să vă ajute)

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:

- Înregistrare live-coding proiect dificultate colocviu/proiect2 – link [aici](#) și pe canalul de Teams. Link proiect din înregistrare [aici](#), pe GitHub.
- Laboratoarele 2-8 care se găsesc [aici](#)

**Important:** În înregistrarea live coding nu am apucat să facem un main() mai complex (sau o clasă principală care servește pe post de main, App spre exemplu, cu metoda Run(), pentru a avea main() minimal). Deși nu se regăsește în înregistrare (e deja destul de lungă și complexă și așa), fac parte din cerință și următoarele: **utilizarea dynamic\_cast** și upcasting/downcasting– a fost ilustrat în repetate rânduri la [Lab06 ex8](#) și [Lab07 ex8](#) (ambele lucrute împreună înregistrate + cod [aici](#) și [aici](#), și pe repository), ierarhie de excepții (am lăsat în cod comentarii cu locuri în care ar putea fi aruncate excepții), [Lab08](#) se ocupă integral de excepții, de asemenea [înregistrat](#).

**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/**, fișiere **.o** în **obj/** și în **.gitignore**. Script de compilare atașat, conform modelului din [Lab03](#) dar și de [aici](#) (cu structura corectă, trebuie adaptat corespunzător proiectului și sistemului de operare)
- Utilizarea corectă a interfețelor, claselor abstracte, mecanismului de excepții.
- Utilizarea corectă de downcasting și ilustrarea pe metode specifice doar acelei derivate

- Documentația codului și comentariile relevante pentru metodele și clasele importante.
- Diagrama cu ierharia de clase rezultată din proiectul vostru

#### IV. Cerințe generale obligatorii

**Observație: pentru a putea fi luat în considerare proiectul pentru notă peste 10, trebuie ca toate cerințele obligatorii să fie implementate**

- O interfață care definește funcțiile virtuale pure
- O clasă abstractă (care implementează anumite metode ale interfeței, dar nu pe toate)
- Variabila și funcție statică
- Variabila constantă
- 2-3 clase care moștenesc clasa abstractă, vector/listă/colecție de pointeri către bază
- Utilizarea corectă a claselor polimorfice, de ilustrat în main() pe o colecție de obiecte de tip Bază 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 Afișare()
- Cel puțin o funcție virtuală care este suprascrisă doar în clasa abstractă și în niciuna din derivate
- 1-2 funcții comune non-virtuale definite în 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 rămâne 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 trebuie să și apelate/testate în cadrul claselor sau în **main()**, altfel nu are sens să le fi definit.
- Fără variabile globale și **fără cale absolută (C:/...)**, doar **cale relativă (.././dir)**
- **FĂRĂ using namespace std** – niciodată în **.h**, tolerabil în **.cpp**, dar tot neindicat; **Observație:** sunteți liberi să îl folosiți la colocviu, unde orice secundă câștigată contează.
- 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. Însă, 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 fii nevoit să introduci 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.