Cerințe etapa 2

Pentru a implementa aceste cerințe, puteți continua/extinde codul de la etapa 1, actualizându-l pentru a folosi conceptele noi.

Proiectul vostru trebuie să respecte în continuare **Criteriile generale** de la prima parte (cod curat, care respectă encapsularea, fără variabile globale, fără clase/-metode care nu sunt folosite/apelate nicăieri), precum și indicațiile referitoare la **Versionarea codului** (continuați să încărcați ce lucrați pe Git) și la **Documentație** (actualizați corespunzător `README`-ul).

Moștenire

- Definiți **minim două ierarhii diferite de moștenire** (două ierarhii de moștenire sunt considerate diferite dacă nu au aceeași clasă de bază și aceeași clasă derivată).
- **Minim o dată membru** și **minim o metodă** care să aibă modificatorul de acces `protected` (în mod util, să fie accesate/apelate dintr-o clasă care le moștenește).
- **Cel puțin o situație** în care să apelați constructorul (cu parametri) al clasei de bază, folosind lista de inițializare din constructorul clasei derivate.

Metode virtuale și clase abstracte

- Definiți și extindeți (moșteniți) **minim o clasă abstractă** (poate avea date membru, dar are cel puțin o metodă pur virtuală).
- Definiți **cel puțin 2 metode virtuale** care să fie suprascrise în clasele moștenitoare. Pot fi pur virtuale sau cu o implementare implicită. Se iau în considerare și metodele definite la celelalte subpuncte, exceptând destructorii virtuali.

Polimorfism la execuție

- Identificați și marcați prin câte un comentariu **minim 2 instanțe** în care să aibă loc polimorfism la execuție (*dynamic dispatch*) în proiectul vostru (e.g. apelul unor metode virtuale prin intermediul unor pointeri/referințe către clasa de bază).
- Identificați și marcați prin câte un comentariu **minim 2 instanțe** de *upcasting* în codul vostru (e.g. atribuirea unor obiecte de tipul unor clase moștenite

la pointeri/referințe către clasa de bază).

Excepții

- Definiți **minim un tip de excepție custom**, care să extindă clasa `exception` din biblioteca standard.
- Aruncați excepții în **minim 2 funcții/metode diferite** (folosiți tipul de excepție definit de voi sau pe cele din biblioteca standard).
- Implementați **minim un bloc** `try...catch` care să prindă o excepție definită și generată de voi (cu specificarea explicită a tipului excepției capturate) și să o trateze într-un fel (să afișeze un mesaj, să reîncerce operațiunea, să arunce o altă excepție etc).

Variabile și metode statice

- Definiți și inițializați o variabilă membru statică în **cel puțin o clasă**.
- Implementați **cel puțin două metode statice** în clasele voastre (din care **cel puțin una** trebuie să acceseze/folosească variabila statică definită la subpunctul anterior).

Bonus

- Separați **declarațiile** și **implementările** din programul vostru folosind fișiere header (`.h`/`.hpp`) și sursă (`.cpp`) distincte. Ar trebui să aveți câte un fișier header și un fișier sursă pentru fiecare clasă din programul vostru.
- Identificați **minim o situație** de moștenire în diamant (trebuie să aibă sens pentru tema voastră). Moștenirea în diamant se referă la a avea o clasă de bază, pe care o moștenesc cu `virtual` două clase distincte, iar apoi aveți o clasă care moștenește ambele clase intermediare.