Laborator 3 – Introducere în Programarea Orientată pe Obiecte

Tema: Studentul Cinefil 🎞

Autor: ing. Eduard Donea

Poveste

Andrei este un student pasionat de filme. În fiecare weekend merge la cinema, ţine un jurnal cu filmele vizionate şi notează cât de mult i-au plăcut. Pentru a-şi organiza mai bine colecţia, decide să scrie un program care să îl ajute să ţină evidenţa filmelor vizionate: titlu, gen, durată şi nota acordată. Astfel, el descoperă conceptul de **clasă** şi **obiect** — primul pas spre Programarea Orientată pe Obiecte.

Scopul laboratorului

Familiarizarea cu noțiunile de:

- definirea unei clase în C++;
- constructori și destructori;
- metode `get/set`;
- alocare dinamică și eliberare de memorie;
- utilizarea `valgrind` pentru verificarea erorilor de memorie.

Structura fișierelor

- **film.h** – declararea clasei `Film`;
- **film.cpp** – implementarea metodelor clasei;
- **main.cpp** – testarea funcționalităților.

Fișierul `film.h`

```
#ifndef FILM_H
#define FILM_H

/* Aici puteti include si bibliotecile pe care le folositi
pentru un cod mai curat si pentru nu a incarca mult main-ul*/

class Film {
//private:
    char* titlu;
    char* gen;
    int durata;   // in minute
    float nota;   // nota acordata de student (0{10)

public:
    // constructor fara parametri
    Film();
    // constructor cu parametri
    Film(const char* t, const char* g, int d, float n);
    // destructor
    ~Film();
    // afiseaza datele filmului
    void afisare() const;
    // modifica nota
    void setNota(float n);
    // returneaza nota
    float getNota() const;
    // returneaza titlul
    const char* getTitlu() const;

};

#endif
```

Fișierul `main.cpp` (model)

```cpp
#include "film.h"

int main() {

    cout << "=== Test individual ===" << '\n';

    // TODO 0: Creati un obiect Film folosind constructorul cu parametri.
    // Exemplu:
    // Film f1("Interstellar", "SF", 169, 9.5);
    // f1.afisare();

    // TODO 1: Testati metodele setNota(), getNota(), getTitlu().
    // Exemplu:
    // f1.setNota(9.8);
    // cout << "Dupa modificare nota: " << f1.getNota() << '\n';
    // cout << "Titlul filmului: " << f1.getTitlu() << '\n';

    cout << "\n=== Vector dinamic ===" << '\n';

    // TODO 2: Cititi numarul de filme si alocati dinamic un vector de pointeri
    ↪  Film.
    // Exemplu de alocare dinamica la vectori de pointeri:
    // Film** v = new Film*[n];

    // TODO 3: Cititi pentru fiecare film titlul, genul, durata si nota.
    // Creati obiecte Film alocate dinamic si salvati-le in vector.

    // TODO 4: Afisati toate filmele introduse apelind metoda afisare() pentru
    ↪  fiecare obiect.

    // TODO 5: Sortati vectorul de pointeri crescator dupa nota filmului.
    // Hint: folositi bubble sort.

    // TODO 6: Determinati si afisati filmul cu nota maxima (filmul preferat).
    // Hint: pastrati indicele obiectului cu cea mai mare nota.

    // TODO 7: Eliberati memoria alocata dinamic,

    // TODO 8: Verificati programul folosind Valgrind pentru scurgeri de
    ↪  memorie.
    // g++ -g main.cpp film.cpp -o lab3
    // valgrind ./lab3

    return 0;
}
```

Cerințe (10p)

1. (4p) Creați o clasă `Film` care să conțină: `titlu`, `gen`, `durata` și `nota`. Implementați:
   - constructor fără parametri;
   - constructor cu parametri;
   - destructor;
   - metode `get`/`set`;
   - metoda `afisare()`.
2. (2p) Creați un vector alocat dinamic de tip `Film`, citiți datele de la tastatură și afișați-le.
3. (1p) Sortați vectorul crescător după `nota` filmului.
4. (1p) Afișați filmul cu cea mai mare `nota`.
5. (1p) Eliberați memoria și verificați cu `valgrind`.
6. (1p) Din oficiu.

Compilare și verificare

```
g++ -g main.cpp film.cpp -o lab3
./lab3
valgrind ./lab3
```

Mesaje utile Valgrind:

- `definitely lost` → memorie ne-eliberată;
- `invalid read/write` → acces invalid în afara zonelor alocate.

Observație: La rularea cu `valgrind`, nu trebuie să existe mesaje de tip `memory leak detected`.

Bonus GitHub Classroom

Studenții pot obține până la **3p bonus** dacă încarcă toate laboratoarele până la finalul semestrului, respectiv **1p bonus** dacă încarcă mai mult de jumătate dintre ele.

Pași pentru încărcarea laboratorului:

1. Accesați cursul de pe OCW – Programare Orientată pe Obiecte, secțiunea **Resurse** → **Bonus GitHub laborator**.
2. Acolo veți găsi linkul **GitHub Classroom** pentru Laboratorul 3:

   `https://classroom.github.com/a/opEG5cqC`

3. După acceptare, veți primi propriul repository, ex. `laborator-3-NumePrenume`.
4. Clonați repository-ul local:

```
git clone
↪  git@github.com:Laborator-POO-2025-2026/laborator-3-NumePrenume.git
cd laborator-3-NumePrenume
```

5. După ce ați adăugat fișierele `film.h`, `film.cpp` și `main.cpp`, urmează comenzile:

```
git add .
git commit -m "Mesaj de commit"
git push origin main
```

Atenție: laboratoarele copiate între studenți sunt considerate nevalide, iar niciunul dintre cei doi participanți nu va primi bonusul. Repository-urile sunt monitorizate automat de sistemul GitHub Classroom.