Laborator 6 – Moștenire simplă

Tema: Student la Poli 🎓

Autor: ing. Eduard Donea

Poveste

Ana, o tânără pasionată de programare, dorește să își construiască o aplicație care să gestioneze datele studenților din facultatea de Automatică și Calculatoare de la UPB. Pentru a înțelege mai bine conceptele de moștenire, ea începe prin a defini o clasă de bază `Persoana`, din care derivă clasa `Student`. În acest mod, fiecare student **este o persoană** (relație *is-a*) și moștenește atributele generale, adăugându-și propriile caracteristici.

Scopul laboratorului

- înțelegerea conceptului de moștenire simplă (`is-a`);
- apelarea constructorului clasei părinte din clasa derivată;
- diferențierea între `private` și `protected`;
- utilizarea operatorilor de flux `<<` și `>>`;
- aplicarea principiilor de reutilizare a codului.

Structura fișierelor

- **persoana.h** – declararea clasei de bază `Persoana`;
- **student.h** – declararea clasei derivate `Student`;
- **main.cpp** – testarea aplicației.

Fișierul `persoana.h`

```h
#ifndef PERSOANA_H
#define PERSOANA_H

#include <iostream>
#include <cstring>
using namespace std;

class Persoana {
protected:
    char* nume;
    int varsta;

public:
    Persoana(const char* n = "Anonim", int v = 0);
    Persoana(const Persoana& p);
    Persoana& operator=(const Persoana& p);
    ~Persoana();

    // Getteri
    const char* getNume() const;
    int getVarsta() const;

    // Setteri
    void setNume(const char* n);
    void setVarsta(int v);

    // Operator de afișare
    friend ostream& operator<<(ostream& out, const Persoana& p);
};
#endif
```

Fișierul `student.h`

```
#ifndef STUDENT_H
#define STUDENT_H

#include "persoana.h"

class Student : public Persoana {
    char* facultate;
    int an;
    float medie;

public:
    Student(const char* n = "Anonim", int v = 0, const char* f = "AC", int a =
    ↪  1, float m = 0.0);
    Student(const Student& s);
    Student& operator=(const Student& s);
    ~Student();

    // Getteri și setteri
    const char* getFacultate() const;
    int getAn() const;
    float getMedie() const;
    void setFacultate(const char* f);
    void setAn(int a);
    void setMedie(float m);

    // Metode suplimentare
    bool esteIntegralist() const; // bonus

    // Operator de afișare
    friend ostream& operator<<(ostream& out, const Student& s);
};
#endif
```

Fișierul `main.cpp` (model)

OBS: Este doar un model, vă puteți scrie main-ul vostru cum doriți.

```cpp
#include "student.h"

int main() {

    // TODO 1: Creati un obiect de tip Student folosind constructorul complet
    Student s1("Ana", 21, "ACS", 2, 8.75);
    cout << s1 << endl;

    // TODO 2: Testati constructorul de copiere
    Student s2(s1);
    cout << s2 << endl;

    // TODO 3: Creati un vector de studenti si cititi datele
    int n;
    cout << "Introduceti numarul de studenti: ";
    cin >> n;
    cin.ignore();

    Student* v = new Student[n];
    for(int i = 0; i < n; i++) {
        char nume[100], fac[100];
        int varsta, an;
        float medie;
        cout << "\nStudent " << i + 1 << ":\n";
        cout << "Nume: "; cin.getline(nume, 100);
        cout << "Varsta: "; cin >> varsta; cin.ignore();
        cout << "Facultate: "; cin.getline(fac, 100);
        cout << "An: "; cin >> an;
        cout << "Medie: "; cin >> medie; cin.ignore();
        v[i] = Student(nume, varsta, fac, an, medie);
    }

    // TODO 4: Afisati toti studentii si media generala
    float suma = 0;
    cout << "\n--- Lista studentilor ---\n";
    for(int i = 0; i < n; i++) {
        cout << v[i] << endl;
        suma += v[i].getMedie();
    }
    cout << "\nMedia generala a grupei: " << suma / n << endl;

    delete[] v;
    return 0;
}
```

Cerințe

1. Implementați clasa `Persoana` cu atributele `nume` și `varsta`.
2. Implementați clasa derivată `Student`, care moștenește `Persoana`.
3. Apelați constructorul clasei părinte în constructorul derivat.
4. Supraincarcați operatorul `<<` pentru afișare.
5. Citiți și afișați o listă de studenți.
6. Calculați media generală a grupei.
7. Bonus: Implementați metoda `esteIntegralist()` care verifică dacă media ≥ 5.

OBS: Notarea la acest laborator va fi înlocuită de rezultatele obținute la testul de la final.

Compilare și rulare

```
g++ -g persoana.cpp student.cpp main.cpp -o lab6
.valgrind /lab6
```

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 6:

   `https://classroom.github.com/a/ZHG6I0Xp`

3. După acceptare, veți primi propriul repository, ex. `laborator-6-NumePrenume`.
4. Clonați repository-ul local:

```
git clone
↪  git@github.com:Laborator-POO-2025-2026/laborator-6-NumePrenume.git
cd laborator-6-NumePrenume
```

5. După ce ați adăugat fișierele `Vehicul.h`, `Vehicul.cpp`, `Masina.h`, `Masina.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.