Laborator 6 – Moștenire simplă

Tema: Phone Master

Autor: ing. Eduard Donea

Poveste

Tudor este pasionat de gadgeturi și dorește să creeze o mică aplicație care gestionează informațiile despre telefoanele sale. Pentru a înțelege mai bine principiul moștenirii, el definește o clasă de bază `Dispozitiv` care conține detalii generale, precum marca și consumul de energie, iar din aceasta derivă clasa `Telefon`, care adaugă informații specifice: capacitatea bateriei și memoria RAM.

Scopul laboratorului

- înțelegerea moștenirii simple (`is-a`);
- apelul constructorului clasei părinte din clasa derivată;
- utilizarea operatorului `<<` pentru afișare;
- diferențierea între `protected` și `private`;
- lucrul cu atribute specifice clasei derivate.

Structura fișierelor

- **dispozitiv.h** – declararea clasei de bază `Dispozitiv`;
- **telefon.h** – declararea clasei derivate `Telefon`;
- **main.cpp** – testarea funcționalităților.

Fișierul `dispozitiv.h`

```h
#ifndef DISPOZITIV_H
#define DISPOZITIV_H

// Includeti bibliotecile necesare aici

class Dispozitiv {
protected:
    char* marca;
    float consum; // in W

public:
    Dispozitiv(const char* m = "Generic", float c = 1.0);
    Dispozitiv(const Dispozitiv& d);
    Dispozitiv& operator=(const Dispozitiv& d);
    ~Dispozitiv();

    // Getteri si setteri
    const char* getMarca() const;
    float getConsum() const;
    void setMarca(const char* m);
    void setConsum(float c);

    // Operator de afisare
    friend ostream& operator<<(ostream& out, const Dispozitiv& d);
};
#endif
```

Fișierul `telefon.h`

```
#ifndef TELEFON_H
#define TELEFON_H

#include "dispozitiv.h"

class Telefon : public Dispozitiv {
    int RAM;        // in GB
    int baterie;    // in mAh

public:
    Telefon(const char* m = "Generic", float c = 1.0, int r = 4, int b = 4000);
    Telefon(const Telefon& t);
    Telefon& operator=(const Telefon& t);
    ~Telefon();

    // Getteri si setteri
    int getRAM() const;
    int getBaterie() const;
    void setRAM(int r);
    void setBaterie(int b);

    // Metode suplimentare
    float autonomie() const; // calculeaza bateria / consum

    // Operator de afisare
    friend ostream& operator<<(ostream& out, const Telefon& t);
};
#endif
```

Fișierul `main.cpp` (model)

OBS: Este doar un exemplu orientativ pentru testare.

```cpp
#include "telefon.h"

int main() {
    // TODO 1: Creati un obiect Telefon
    Telefon t1("Samsung", 2.5, 8, 5000);
    cout << t1 << endl;

    // TODO 2: Creati un vector de telefoane
    int n;
    cout << "\nIntroduceti numarul de telefoane: ";
    cin >> n;
    cin.ignore();

    Telefon* v = new Telefon[n];
    for (int i = 0; i < n; i++) {
        char marca[50];
        float consum;
        int ram, baterie;

        cout << "\nTelefon " << i + 1 << ":\n";
        cout << "Marca: "; cin.getline(marca, 50);
        cout << "Consum (W): "; cin >> consum;
        cout << "RAM (GB): "; cin >> ram;
        cout << "Baterie (mAh): "; cin >> baterie;
        cin.ignore();

        v[i] = Telefon(marca, consum, ram, baterie);
    }

    // TODO 3: Afisati telefoanele cu baterie > 4000mAh
    cout << "\nTelefoane cu baterie > 4000mAh:\n";
    for (int i = 0; i < n; i++)
        if (v[i].getBaterie() > 4000)
            cout << v[i] << endl;

    // TODO 4: Afisati consumul mediu
    float total = 0;
    for (int i = 0; i < n; i++)
        total += v[i].getConsum();
    cout << "\nConsum mediu: " << total / n << " W" << endl;

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

Cerințe

1. Implementați clasa `Dispozitiv` cu atributele `marca` și `consum`.
2. Implementați clasa derivată `Telefon` care moștenește `Dispozitiv`.
3. Supraîncarcați operatorul `<<` pentru afișare.
4. Citiți și afișați un vector de telefoane și filtrați-le după capacitatea bateriei.
5. Afișați consumul mediu total.
6. Bonus: Implementați metoda `autonomie()` care returnează `baterie / consum`.

Compilare și rulare

```
g++ -g dispozitiv.cpp telefon.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.