Laborator 8 – Funcții și clase template

Tema: Mini Discord – Profile Manager 💬

Autor: ing. Eduard Donea

Poveste

Radu, un pasionat de jocuri și comunități online, a decis să creeze propria platformă de chat în stil Discord, unde utilizatorii sunt reprezentați prin profiluri digitale. Aceștia pot fi utilizatori obișnuiți, moderatori sau chiar roboți inteligenți. Pentru a gestiona aceste entități într-un mod generic și eficient, Radu folosește `template`-uri pentru a crea un manager de profiluri reutilizabil.

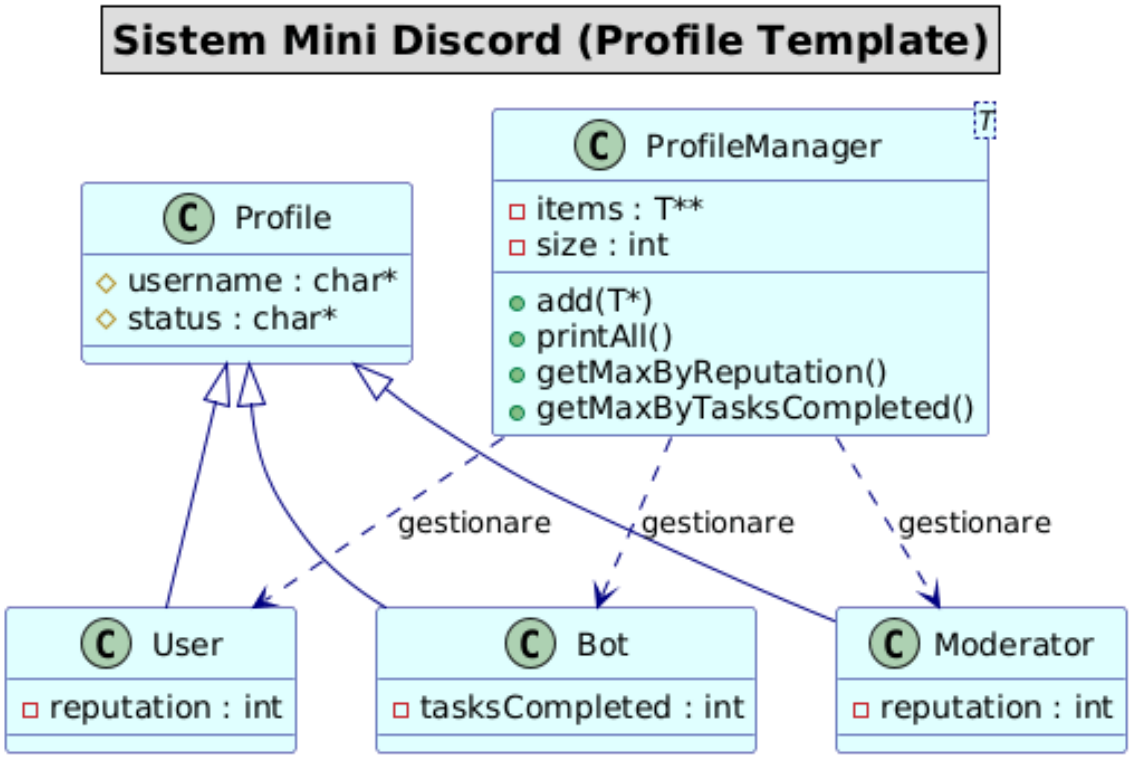
Scopul laboratorului

- înțelegerea conceptului de **template** la nivel de clasa;
- definirea și instanțierea explicită a unui template;
- aplicarea moștenirii și agregării simple;
- organizarea corectă a fișierelor într-un proiect C++.

Structura fișierelor

- **Profile.h** – clasa de bază pentru profiluri;
- **User.h** – clasa derivată pentru utilizatorii obișnuiți;
- **Moderator.h** – clasa derivată pentru moderatori;
- **Bot.h** – clasa derivată pentru boti;
- **ProfileManager.h** – definiția clasei template;

Figura 1 – Diagrama sistemului Mini Discord



În continuare aveți scheletele claselor. **Atenție: metodele trebuie implementate de voi acolo unde lipsesc. La template, aveți .h, iar la .cpp aveți doar un model de implementare pentru o metodă.**

Fișierul `Profile.h`

```
#ifndef PROFILE_H
#define PROFILE_H

// Includeti aici bibliotecile necesare

class Profile {
protected:
    char* username;
    char* status;

public:
    // TODO: Scieti aici tot ceea ce aveti nevoie.
};

#endif
```

Fișierul `User.h`

```
#ifndef USER_H
#define USER_H

#include "Profile.h"

class User : public Profile {
private:
    int reputation;

public:
    // TODO: Scieti aici tot ceea ce aveti nevoie.
};

#endif
```

Fișierul `Moderator.h`

```
#ifndef MODERATOR_H
#define MODERATOR_H

#include "Profile.h"

class Moderator : public Profile {
private:
    int reputation;

public:
    // TODO: Scieti aici tot ceea ce aveti nevoie.
};

#endif
```

Fișierul `Bot.h`

```
#ifndef BOT_H
#define BOT_H

#include "Profile.h"

class Bot : public Profile {
private:
    int tasksCompleted;

public:
    // TODO: Scieti aici tot ceea ce aveti nevoie.
};

#endif
```

Fișierul `ProfileManager.h`

```h
#ifndef PROFILEMANAGER_H
#define PROFILEMANAGER_H

#include "User.h"
#include "Moderator.h"
#include "Bot.h"

template <class T>
class ProfileManager {
private:
    T** items;
    int size;

public:
    ProfileManager();
    ~ProfileManager();

    void add(T* item);
    void printAll() const;

    T* getMaxByReputation();        // pentru User, Moderator
    T* getMaxByTasksCompleted();    // pentru Bot

};

void test();                        // Functie care ajuta la linker

#endif
```

Fișierul `ProfileManager.cpp` - model de implementare

```cpp
#include "ProfileManager.h"

template <class T>
ProfileManager<T>::ProfileManager() : size(0), items(nullptr) {}

// Mereu sa aveti grija sa puneti "template <class T>" la
// inceput de orice metoda/constructor template.

// Nu uitati sa va scieti void test(), inclunzand inauntrul ei
// tot ce faceti in "int main()", pana la "return 0", nu includeti si asta.
```

Cerințe (10p)

1. (3p) Implementați clasele `Profile`, `User`, `Moderator` și `Bot`, completând metodele lipsă;
2. (3p) Implementați clasa template `ProfileManager`;
3. (2p) Testați funcțiile template în `main.cpp`;
4. (1p) Utilizați `getMaxByReputation()` și `getMaxByTasksCompleted()`;
5. (1p) Eliberați memoria și verificați cu `valgrind`.

Compilare si rulare

```
g++ -g *.cpp -o lab8
valgrind ./lab8
```

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 8:

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

3. După acceptare, veți primi propriul repository, ex. `laborator-8-NumePrenume`.
4. Clonați repository-ul local:

```
git clone
↪  git@github.com:Laborator-POO-2025-2026/laborator-8-NumePrenume.git
cd laborator-8-NumePrenume
```

5. După ce ați adăugat fișierele, 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.