Laborator 8 – Funcții și clase template

Tema: RPG Inventory Manager

Autor: ing. Eduard Donea

Poveste

Alex este un tânaăr programator pasionat de dezvoltarea de jocuri video. După ce a înăațat bazele programării orientate pe obiecte, el vrea acum să implementeze un sistem modular pentru gestionarea inventarului unui joc RPG. În acest sistem, fiecare obiect poate fi de un anumit tip (armă, poțiune etc.), iar inventarul trebuie să fie generic și reutilizabil prin intermediul `template`-urilor.

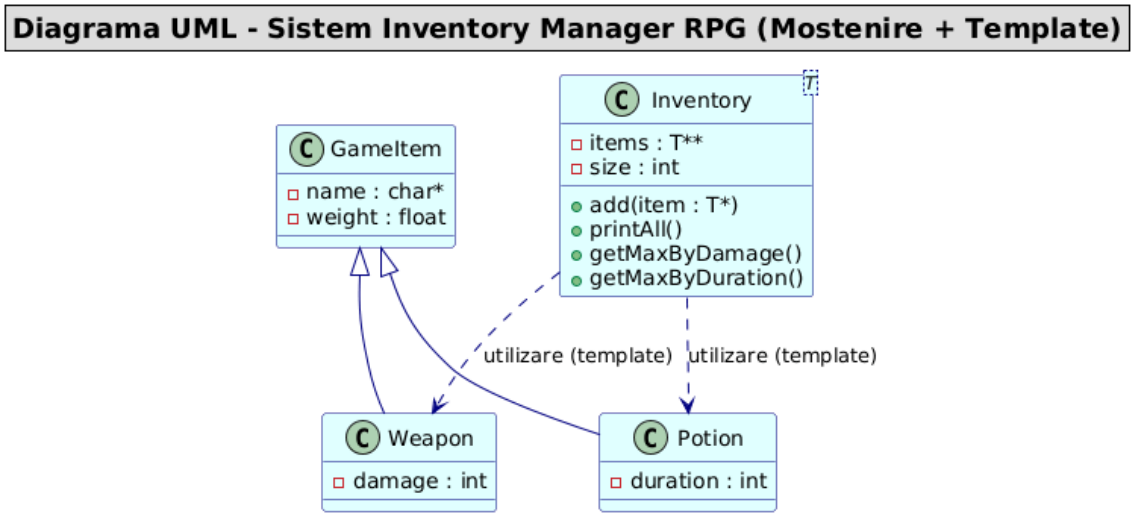
Scopul laboratorului

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

Structura fișierelor

- **GameItem.h** – clasa de bază;
- **Weapon.h** – clasa derivată din `GameItem`;
- **Potion.h** – clasa derivată din `GameItem`;
- **Inventory.h** – definiția clasei template;

Figura 1 – Diagrama UML a sistemului RPG Inventory Manager



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 `GameItem.h`

```h
#ifndef GAMEITEM_H
#define GAMEITEM_H

// Includeti aici bibliotecile necesare

class GameItem {
protected:
    char* name;
    float weight;

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

};

#endif
```

Fișierul `Weapon.h`

```h
#ifndef WEAPON_H
#define WEAPON_H

#include "GameItem.h"

class Weapon : public GameItem {
private:
    int damage;

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

#endif
```

Fișierul `Potion.h`

```h
#ifndef POTION_H
#define POTION_H

#include "GameItem.h"

class Potion : public GameItem {
private:
    int duration;

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

};

#endif
```

Fișierul `Inventory.h`

```h
#ifndef INVENTORY_H
#define INVENTORY_H

#include "Weapon.h"
#include "Potion.h"

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

public:
    Inventory();
    ~Inventory();

    void add(T* item);
    void printAll() const;
    T* getMaxByDamage();    // pentru Weapon
    T* getMaxByDuration();  // pentru Potion
};

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

#endif
```

Fișierul `inventory.cpp` - model de implementare

```cpp
#include "Inventory.h"

template <class T>
Inventory<T>::Inventory() {
    size = 0;
    items = nullptr;
}

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

// 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 `GameItem`, `Weapon` si `Potion`, completând metodele lipsă (constructori, set/get etc.)..
2. (3p) Implementați clasa template `Inventory`.
3. (2p) Testați funcțiile template în `main.cpp`.
4. (1p) Utilizați `getMaxByDamage()` și `getMaxByDuration()`.
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.