Laborator POO - Metin2: Remastered

322ABb

03/12/2024

Introducere

Mircea, pasionat de jocurile RPG, s-a hotărât să reinventeze jocul său preferat, Metin2. Prin viziunea sa modernă și pasiunea pentru programare, Mircea a creat o nouă versiune intitulată **Metin2: Remastered**. Aceasta aduce o experiență captivantă de luptă între echipe de caractere, fiecare cu abilități speciale unice.

Proiectul are ca scop aplicarea conceptelor avansate de programare orientată pe obiect, incluzând:

- Vector neomogen.
- Clase abstracte și moștenire.
- Polimorfism.
- Gestionarea dinamică a memoriei.

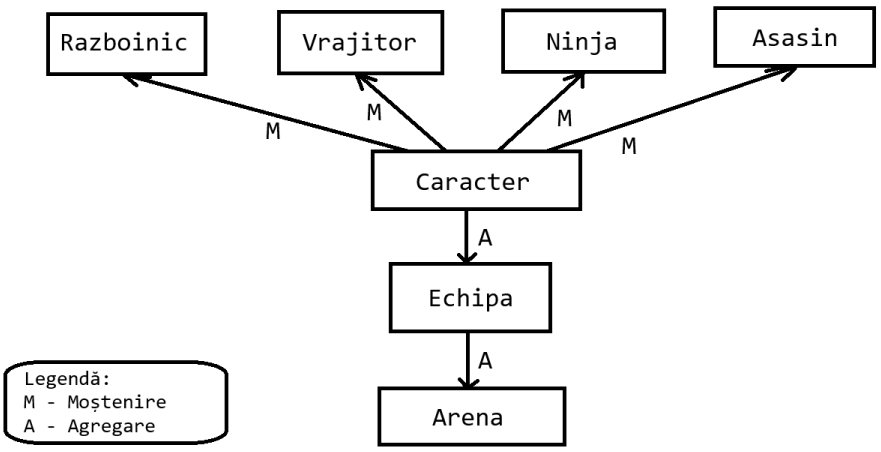
Structura Proiectului

Proiectul este compus din următoarele fișiere:

- **Caracter.h** și **Caracter.cpp** - Definirea clasei abstracte de bază pentru caractere.
- **Razboinic.h/cpp**, **Vrajitor.h/cpp**, **Ninja.h/cpp**, **Asasin.h/cpp** - Clase derivate care implementează caracterele.
- **Echipa.h/cpp** - Gestionarea echipei printr-un vector dinamic.
- **Arena.h/cpp** - Simularea luptei și salvarea logului.
- **main.cpp** - Punctul de pornire al aplicației.

Schema Proiectului

Mai jos este schema completă a claselor din proiect. Aceasta ilustrează relațiile de moștenire și agregare între clase:



Codul Proiectului

Aici sunt prezentate fișierele principale ale proiectului. Completați zonele marcate cu `TODO` dacă este necesar. Veți avea și fișierele text cu headerele și sursele.

Caracter.h

```
#ifndef CARACTER_H
#define CARACTER_H

#include <iostream>
#include <cstring>

class Caracter {
protected:
    char* nume;
    int viata;
    int atac;
    int aparare;

public:
    Caracter(const char* nume, int viata, int atac, int aparare); // TODO:
        Implementati constructorul
    virtual ~Caracter(); // TODO: Implementati destructorul

    virtual void folosesteAbilitate() = 0; // TODO: Clasele derivate vor defini
         aceasta metoda
    virtual void afisareDetalii(std::ostream& out) const = 0;

    friend std::ostream& operator<<(std::ostream& out, const Caracter& c);

    bool esteEliminat() const; // TODO: Implementati verificarea eliminarii
    void primesteDaune(int daune); // TODO: Calculati daunele primite

    int getAtac() const; // TODO: Implementati getter
    const char* getName() const; // TODO: Implementati getter
};

#endif // CARACTER_H
```

Razboinic.h

```
#ifndef RAZBOINIC_H
#define RAZBOINIC_H

#include "Caracter.h"
#include <iostream>

class Razboinic : public Caracter {
public:
    Razboinic(const char* nume);
    void folosesteAbilitate() override;
    void afisareDetalii(std::ostream& out) const override;
};

#endif // RAZBOINIC_H
```

Razboinic.cpp

```
#include "Razboinic.h"

Razboinic::Razboinic(const char* nume) : Caracter(nume, 150, 50, 30) {}

void Razboinic::folosesteAbilitate() {

    atac += 15;
    std::cout << nume << " foloseste furia razboinicului! Atacul creste la " <<
         atac << ".\n";

}

void Razboinic::afisareDetalii(std::ostream& out) const {

    out << "Razboinic: " << nume << " | Viata: " << viata
        << " | Atac: " << atac << " | Aparare: " << aparare << "\n";

}
```

TODO: Restul claselor derivate (Vrajitor, Ninja, Asasin), asemănător cu clasa Razboinic.

Echipa.h

```
#ifndef ECHIPA_H
#define ECHIPA_H

#include "Caracter.h"
#include <iostream>

class Echipa {
private:
    Caracter** membri;
    int capacitate;
    int nrMembri;

public:
    Echipa(int capacitateMax = 10);
    ~Echipa();

    void adaugaCaracter(Caracter* c);
    void afiseazaEchipa() const;
    bool esteInvinsa() const;

    Caracter* getCaracterNeeliminat();
};

#endif // ECHIPA_H
```

Echipa.cpp

```
#include "Echipa.h"

Echipa::Echipa(int capacitateMax) {
    // TODO: Implementare
}

Echipa::~Echipa() {
    // TODO: Implementare
}

void Echipa::adaugaCaracter(Caracter* c) {

    // TODO: Implementare

}

void Echipa::afiseazaEchipa() const {

    for (int i = 0; i < nrMembri; i++) {

        //std::cout << *membri[i]; // Afisare normala

        TODO: Faceti o afisare detaliata folosinf metoda afisareDetalii()

}

bool Echipa::esteInvinsa() const {

    for (int i = 0; i < nrMembri; i++) {

        if (!membri[i]->esteEliminat())
            return false;

    }

    return true;

}

Caracter* Echipa::getCaracterNeeliminat() {

    for (int i = 0; i < nrMembri; i++) {

        if (!membri[i]->esteEliminat())
     return membri[i];

    }

    return nullptr;

}
```

Arena.h

```
#ifndef ARENA_H
#define ARENA_H

#include "Echipa.h"

#include "Razboinic.h"
#include "Vrajitor.h"
#include "Ninja.h"
#include "Asasin.h"

#include <iostream>

class Arena {
private:
    Echipa echipa1;
    Echipa echipa2;

public:
    Arena(Echipa& e1, Echipa& e2);

    void simuleazaLupta();
    void afiseazaRezultate() const;
};

#endif // ARENA_H
```

Arena.cpp

```
#include "Arena.h"

Arena::Arena(Echipa& e1, Echipa& e2) : echipa1(e1), echipa2(e2) {}

void Arena::simuleazaLupta() {
    bool echipa1Ataca = true; // Indica cine ataca in acest tur

    while (!echipa1.esteInvinsa() && !echipa2.esteInvinsa()) {
        if (echipa1Ataca) {
            Caracter* atacant1 = echipa1.getCaracterNeeliminat();
            Caracter* atacant2 = echipa2.getCaracterNeeliminat();

            if (atacant1 && atacant2) {
                atacant1->folosesteAbilitate();
                int damage1 = atacant1->getAtac();
                atacant2->primesteDaune(damage1);

                std::cout << atacant1->getName() << " ataca pe " << atacant2->
                    getName()
                    << " si provoaca " << damage1 << " daune.\n";

                if (atacant2->esteEliminat()) {
                    std::cout << atacant2->getName() << " a fost eliminat!\n";
                }
            }
        }
        else {
            Caracter* atacant2 = echipa2.getCaracterNeeliminat();
            Caracter* atacant1 = echipa1.getCaracterNeeliminat();

            if (atacant2 && atacant1) {
                atacant2->folosesteAbilitate();
                int damage2 = atacant2->getAtac();
                atacant1->primesteDaune(damage2);

                std::cout << atacant2->getName() << " ataca pe " << atacant1->
                    getName()
                    << " si provoaca " << damage2 << " daune.\n";

                if (atacant1->esteEliminat()) {
                    std::cout << atacant1->getName() << " a fost eliminat!\n";
                }
            }
        }

        // Schimba randul de atac
        echipa1Ataca = !echipa1Ataca;
    }
}


void Arena::afiseazaRezultate() const {
    if (echipa1.esteInvinsa()) {
        std::cout << "Echipa 2 castiga!\n";
    }
    else {
        std::cout << "Echipa 1 castiga!\n";
    }
}
```

main.cpp

```cpp
#inlcude "Arena.h"

int main() {

    // TODO: Configurati echipele si rulati simularea luptei

    return 0;

}
```

Punctaj

Proiectul este evaluat pe baza următoarelor criterii, pentru un total de 10 puncte:

- **Implementare clase derivate noi (maxim 4 să fie în total)** - 3p
- **Implementare completă pentru toate clasele derivate** - 2p: Scrieți constructorii, metodele 'folosesteAbilitate' și afișările pentru toate clasele.
- **Gestionarea echipei** - 2p: Completați logica adăugării de caractere și afișării membrilor echipei.
- **Implementare main.cpp** - 2p
- **Oficiu** - 1p