Лабораторная работа № 1.3

Тема: Разработка текстовой RPG-игры. Полиморфизм

#include <iostream>

#include <string>

#include <cstdlib>

#include <ctime>

class Entity {

protected:

std::string name;

int health;

int attackPower;

int defense;

int maxHealth; // Для метода heal

public:

Entity(const std::string& n, int h, int a, int d)

: name(n), health(h), attackPower(a), defense(d), maxHealth(h) {}

int getDefense() const { return defense; }

int getHealth() const { return health; }

std::string getName() const { return name; }

void setHealth(int h) { health = std::min(h, maxHealth); }

// Виртуальный метод для атаки

virtual void attack(Entity& target) {

int damage = attackPower - target.defense;

if (damage > 0) {

target.health -= damage;

std::cout << name << " attacks " << target.name << " for " << damage << " damage!\n";

} else {

std::cout << name << " attacks " << target.name << ", but it has no effect!\n";

}

}

// Виртуальный метод для вывода информации

virtual void displayInfo() const {

std::cout << "Name: " << name << ", HP: " << health

<< ", Attack: " << attackPower << ", Defense: " << defense << std::endl;

}

// Метод heal

virtual void heal(int amount) {

health = std::min(health + amount, maxHealth);

std::cout << name << " healed for " << amount << " HP!\n";

}

// Виртуальный деструктор

virtual ~Entity() {}

};

class Monster : public Entity {

public:

Monster(const std::string& n, int h, int a, int d)

: Entity(n, h, a, d) {}

// Переопределение метода attack

void attack(Entity& target) override {

int damage = attackPower - target.getDefense();

if (damage > 0) {

// Шанс на ядовитую атаку (30%)

if (rand() % 100 < 30) {

damage += 5; // Дополнительный урон от яда

std::cout << "Poisonous attack! ";

}

target.setHealth(target.getHealth()- damage);

std::cout << name << " attacks " << target.getName() << " for " << damage << " damage!\n";

} else {

std::cout << name << " attacks " << target.getName() << ", but it has no effect!\n";

}

}

// Переопределение метода displayInfo

void displayInfo() const override {

std::cout << "Monster: " << name << ", HP: " << health

<< ", Attack: " << attackPower << ", Defense: " << defense << std::endl;

}

};

// Класс босс с уникальной способностью

class Boss : public Monster {

private:

std::string specialAbility;

public:

Boss(const std::string& n, int h, int a, int d, const std::string& ability)

: Monster(n, h, a, d), specialAbility(ability) {}

// Переопределение метода attack

void attack(Entity& target) override {

Monster::attack(target); // Базовая атака Monster

if (rand() % 100 < 25) { // 25% шанс на огненный удар

int fireDamage = 10;

target.setHealth(target.getHealth()- fireDamage);

std::cout << specialAbility << "! " << name

<< " deals additional " << fireDamage << " fire damage!\n";

}

}

void displayInfo() const override {

std::cout << "Boss: " << name << ", HP: " << health

<< ", Ability: " << specialAbility << std::endl;

}

};

class Character : public Entity {

public:

Character(const std::string& n, int h, int a, int d)

: Entity(n, h, a, d) {}

// Переопределение метода attack

void attack(Entity& target) override {

int damage = attackPower - target.getDefense();

if (damage > 0) {

// Шанс на критический удар (20%)

if (rand() % 100 < 20) {

damage \*= 2;

std::cout << "Critical hit! ";

}

target.setHealth(target.getHealth() - damage);

std::cout << name << " attacks " << target.getName() << " for " << damage << " damage!\n";

} else {

std::cout << name << " attacks " << target.getName() << ", but it has no effect!\n";

}

}

// Переопределение метода heal

void heal(int amount) override {

Entity::heal(amount); // Используем базовую реализацию

std::cout << name << "'s current HP: " << health << "/" << maxHealth << "\n";

}

// Переопределение метода displayInfo

void displayInfo() const override {

std::cout << "Character: " << name << ", HP: " << health

<< ", Attack: " << attackPower << ", Defense: " << defense << std::endl;

}

};

int main() {

srand(static\_cast<unsigned>(time(0))); // Инициализация генератора случайных чисел

// Создание объектов

Character hero("Hero", 100, 20, 10);

Boss dragon("Dragon", 500, 30, 20, "Inferno Breath");

// Демонстрация метода heal

hero.displayInfo();

hero.heal(50); // Лечение

hero.displayInfo();

// Полиморфное поведение

Entity\* entities[] = { &hero, &dragon };

for (auto& entity : entities) {

entity->displayInfo(); // Вывод информации о сущности

}

// Бой

hero.attack(dragon);

dragon.attack(hero);

return 0;

}