Лабораторная работа № 1.2

Тема: Разработка текстовой RPG-игры. Наследование

#include <iostream>

#include <string>

class Entity {

protected:

std::string name; // Защищенное поле: имя

int health; // Защищенное поле: здоровье

public:

// Конструктор базового класса

Entity(const std::string& n, int h) : name(n), health(h) {}

// Метод для вывода информации

virtual void displayInfo() const {

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

}

virtual ~Entity() {}

};

class Player : public Entity {

private:

int experience; // Приватное поле: опыт

public:

// Конструктор производного класса

Player(const std::string& n, int h, int exp)

: Entity(n, h), experience(exp) {}

// Переопределение метода displayInfo

void displayInfo() const override {

Entity::displayInfo(); // Вызов метода базового класса

std::cout << "Experience: " << experience << std::endl;

}

};

class Enemy : public Entity {

private:

std::string type; // Приватное поле: тип врага

public:

// Конструктор производного класса

Enemy(const std::string& n, int h, const std::string& t)

: Entity(n, h), type(t) {}

// Переопределение метода displayInfo

void displayInfo() const override {

Entity::displayInfo(); // Вызов метода базового класса

std::cout << "Type: " << type << std::endl;

}

};

class Boss : public Enemy {

private:

std::string specialAbility; // Приватное поле: способность босса

public:

// Конструктор производного класса

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

: Enemy(n, h, t), specialAbility(ability) {}

// Переопределение метода displayInfo()

void displayInfo() const override {

Enemy::displayInfo(); // Вызов метода Enemy

std::cout << "Special Ability: " << specialAbility << std::endl;

}

};

int main() {

// Создаем объект босс

Boss finalBoss("Dark Dragon", 500, "Dragon", "Inferno Flame");

// Выводим информацию о боссе

finalBoss.displayInfo();

return 0;

}