МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ

ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ

УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

«БЕЛГОРОДСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНОЛОГИЧЕСКИЙ

УНИВЕРСИТЕТ им. В. Г. ШУХОВА» (БГТУ им. В.Г. Шухова)

Кафедра программного обеспечения

вычислительной техники и автоматизированных

систем

**Лабораторная работа №2**

по дисциплине: ООП

тема: «Модульное программирование. Интерфейсы»

Выполнил: студент группы ВТ-231

Самойленко Кирилл

Проверили:  
Буханов Д. Г.

Белгород 2025

**Цель работы:**Получение навыков модульной декомпозиции предметной области,

создания модулей. Разработка интерфейсов.

Вариант 17  
  
**Файл Interface.h**

#pragma once

#ifndef INTERFACES\_H

#define INTERFACES\_H

#include <locale.h>

#include <iostream>

#include <string>

#include <vector>

class Ball;

class IPlayer {

public:

virtual void kick(Ball& ball) = 0;

virtual void save() = 0;

virtual std::string getName() const = 0;

virtual std::string getPosition() const = 0;

};

class ITeam {

public:

virtual std::string getName() const = 0;

virtual const std::vector<IPlayer\*>& getPlayers() const = 0;

};

class IStadium {

public:

virtual std::string getName() const = 0;

virtual int getCapacity() const = 0;

};

class IReferee {

public:

virtual void startMatch() = 0;

virtual std::string getName() const = 0;

};

class IBall {

public:

virtual std::string getBrand() const = 0;

virtual double getWeight() const = 0;

};

class Player : public IPlayer {

public:

Player(const std::string& name, const std::string& position);

void kick(Ball& ball) override;

void save() override;

std::string getName() const override;

std::string getPosition() const override;

private:

std::string name;

std::string position;

};

class Team : public ITeam {

public:

Team(const std::string& name, const std::vector<IPlayer\*>& players);

std::string getName() const override;

const std::vector<IPlayer\*>& getPlayers() const override;

private:

std::string name;

std::vector<IPlayer\*> players;

};

class Stadium : public IStadium {

public:

Stadium(const std::string& name, int capacity);

std::string getName() const override;

int getCapacity() const override;

private:

std::string name;

int capacity;

};

class Referee : public IReferee {

public:

Referee(const std::string& name);

void startMatch() override;

std::string getName() const override;

private:

std::string name;

};

class Ball : public IBall {

public:

Ball(const std::string& brand, double weight);

std::string getBrand() const override;

double getWeight() const override;

private:

std::string brand;

double weight;

};

#endif // INTERFACES\_H

**Файл interface.cpp**

#include "interfaces.h"

#include <locale.h>

#include <iostream>

Player::Player(const std::string& name, const std::string& position)

: name(name), position(position) {}

void Player::kick(Ball& ball) {

std::cout << name << " ударил по мячу " << ball.getBrand() << "!\n";

}

void Player::save() {

std::cout << name << " спас ворота!\n";

}

std::string Player::getName() const {

return name;

}

std::string Player::getPosition() const {

return position;

}

Team::Team(const std::string& name, const std::vector<IPlayer\*>& players)

: name(name), players(players) {}

std::string Team::getName() const {

return name;

}

const std::vector<IPlayer\*>& Team::getPlayers() const {

return players;

}

Stadium::Stadium(const std::string& name, int capacity)

: name(name), capacity(capacity) {}

std::string Stadium::getName() const {

return name;

}

int Stadium::getCapacity() const {

return capacity;

}

Referee::Referee(const std::string& name)

: name(name) {}

void Referee::startMatch() {

std::cout << "Судья " << name << " дает свисток к началу матча!\n";

}

std::string Referee::getName() const {

return name;

}

Ball::Ball(const std::string& brand, double weight)

: brand(brand), weight(weight) {}

std::string Ball::getBrand() const {

return brand;

}

double Ball::getWeight() const {

return weight;

}

### Файл Main.cpp

#include <iostream>

#include "interfaces.h"

#include <locale.h>

int main() {

setlocale(LC\_ALL, "Rus");

Player player1("Лев Яшин", "вратарь");

Player player2("Пеле", "нападающий");

Player player3("Игорь Акинфеев", "вратарь");

Player player4("Артем Дзюба", "нападающий");

Team team1("Сборная Бразилии", { &player1, &player2 });

Team team2("Сборная России", { &player3, &player4 });

Stadium stadium("Лужники", 81000);

Referee referee("Пьерлуиджи Коллина");

Ball ball("Nike", 0.7);

std::cout << "Матч начинается на стадионе " << stadium.getName() << "!\n";

std::cout << "Судья: " << referee.getName() << "\n";

std::cout << "Команды: " << team1.getName() << " vs " << team2.getName() << "\n";

referee.startMatch();

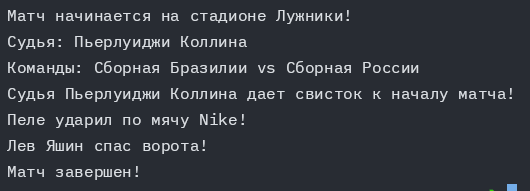
player2.kick(ball);

player1.save();

std::cout << "Матч завершен!\n";

return 0;

}



**Вывод**: получили навыки модульной декомпозиции предметной области, создания моду-

лей и разработали интерфейс по условию задачи.