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: Объектно-ориентированное программирование Тема: Модульное программирование. Интерфейсы.

Выполнил: студент группы ПВ-223

Дмитриев Андрей Александрович

Проверил: Черников С.В.

Белгород 2024 г.

**Цель работы**: Получение навыков модульной декомпозиции предметной области, создания модулей. Разработка интерфейсов.

**Задание**: Разработать программу, состоящую из трех модулей в соответствии с указанным вариантом задания. Первый модуль – основной код программы; второй содержит интерфейсы; третий модуль – реализацию этих интерфейсов. Количество структур данных ("объектов") не менее пяти.

**Задание варианта**: Программа «Домашняя аудио-коллекция»

# Модуль main.cpp

#include <iostream>

#include "Headers/AudioCollection.h"

int main()

{

    Author a;

    a.name = "puper author";

    a.positionInTop = 1;

    Track t1;

    t1.author = a;

    t1.name = "test";

    t1.duration = 400;

    t1.textOfTrack = "testtesttest";

    Track t2;

    t2.author = a;

    t2.name = "never fade away";

    t2.duration = 18342;

    t2.textOfTrack = "f arasaka";

    Album al1;

    al1.name = "empty album";

    al1.totalDuration = 0;

    al1.tracks.push\_back(t2);

    al1.tracks.push\_back(t1);

    AudioCollection ac1;

    ac1.add(t1);

    ac1.add(t2);

    ac1.add(al1);

    Player p;

    for (Track t : ac1.tracks)

        p.addTrackToPlayList(t);

    p.play();

    std::cout << p.getCurrentTrack().name << " -> " << p.next().name << '\n';

    p.putAlbumToPlaylist(al1);

    std::cout << p.getCurrentTrack().name << " -> " << p.next().name;

}

# Модуль AudioCollection.h

#ifndef AudioCollection\_h

#define AudioCollection\_h

#include <string>

#include <list>

#include <vector>

struct Author{

    std::string name;

    int positionInTop;

};

struct Track{

    Author author;

    std::string name;

    int duration;

    std::string textOfTrack;

};

struct Album{

    std::list<Track> tracks;

    std::string name;

    int totalDuration;

};

struct AudioCollection{

    std::list<Track> tracks;

    std::list<Album> albums;

    void add(Track track);

    void add(Album album);

};

enum PlayerState {

    PLAY,

    STOP

};

struct Player{

    AudioCollection audioCollection;

    std::vector<Track> playlist;

    int indexCurrentTrack;

    PlayerState state;

    void addTrackToPlayList(Track track);

    void putAlbumToPlaylist(Album album);

    Track getCurrentTrack();

    Track play();

    Track stop();

    Track next();

    Track prev();

};

#endif

# Модуль AudioCollection.cpp

#include "../Headers/AudioCollection.h"

void AudioCollection::add(Track track)

{

    tracks.push\_back(track);

}

void AudioCollection::add(Album album)

{

    albums.push\_back(album);

}

# Модуль Player.cpp

#include "../Headers/AudioCollection.h"

void Player::addTrackToPlayList(Track track)

{

    playlist.push\_back(track);

}

void Player::putAlbumToPlaylist(Album album)

{

    playlist.clear();

    indexCurrentTrack = 0;

    for (Track track : album.tracks)

        addTrackToPlayList(track);

}

Track Player::play()

{

    state = PLAY;

    if (indexCurrentTrack < 0)

        indexCurrentTrack = 0;

    return playlist[indexCurrentTrack];

}

Track Player::stop()

{

    state = STOP;

    return playlist[indexCurrentTrack];

}

Track Player::next()

{

    if (indexCurrentTrack < playlist.size() - 1)

        indexCurrentTrack++;

    return playlist[indexCurrentTrack];

}

Track Player::prev()

{

    if (indexCurrentTrack > 1)

        indexCurrentTrack--;

    return playlist[indexCurrentTrack];

}

Track Player::getCurrentTrack()

{

    if (indexCurrentTrack < 0

        && indexCurrentTrack >= playlist.size())

        throw;

    return playlist[indexCurrentTrack];

}

# Результат работы:

# 

**Вывод:** Получили навыки модульной декомпозиции предметной области. Разработали интерфейс.