|  |
| --- |
|  |
| Object Oriented Programming |
| Project Report |
| **Course Instructor:** Asst Prof Anum Abdul Salam  **Lab Engineer:** Lab Engineer Sundus Ashraf  **Student Name:** Amina Qadeer  **Registration Number:** 359607  **Degree/ Syndicate:** CE 42 A |

DATE:

30th January, 2022

## Code

#include<iostream>

#include<string>

#include<dirent.h>

#include<stack>

#include<Windows.h>

#include<mmsystem.h>

#include<direct.h>

#include<algorithm>

#include <random>

using namespace std;

wstring s2ws(const std::string& s)

{

int len;

int slength = (int)s.length() + 1;

len = MultiByteToWideChar(CP\_ACP, 0, s.c\_str(), slength, 0, 0);

wchar\_t\* buf = new wchar\_t[len];

MultiByteToWideChar(CP\_ACP, 0, s.c\_str(), slength, buf, len);

std::wstring r(buf);

delete[] buf;

return r;

}

class StorageMedia {

public:

virtual string\* getAudios(int& x) {};

virtual string getDirectory()

{

return "";

}

};

class USB : public StorageMedia {

string directory;

public:

USB(string directory)

{

this->directory = directory;

}

string\* getAudios(int& x)

{

stack <string> audi;

DIR\* d;

struct dirent\* dir;

d = opendir(directory.c\_str());

if (d)

{

while ((dir = readdir(d)) != NULL)

{

audi.push(dir->d\_name);

}

closedir(d);

}

x = audi.size();

string\* aud = new string[audi.size()];

for (int i = 0; i < audi.size(); i++)

{

aud[i] = audi.top();

audi.pop();

}

return aud;

}

string getDirectory()

{

return directory;

}

};

class CD : public StorageMedia {

string directory;

public:

CD(string directory)

{

this->directory = directory;

}

string\* getAudios(int& x)

{

stack <string> audi;

DIR\* d;

struct dirent\* dir;

d = opendir(directory.c\_str());

if (d)

{

while ((dir = readdir(d)) != NULL)

{

audi.push(dir->d\_name);

}

closedir(d);

}

x = audi.size();

string\* aud = new string[audi.size()];

for (int i = 0; i < audi.size(); i++)

{

aud[i] = audi.top();

audi.pop();

}

return aud;

}

string getDirectory()

{

return directory;

}

};

class AudioPlayer {

string manufacturerName;

string model;

public:

AudioPlayer()

{

manufacturerName = "zyx";

model = "2022";

}

};

class InCar : public AudioPlayer {

protected:

StorageMedia\* m;

bool state;

bool mount;

string\* audios;

int TotalSongs;

public:

InCar()

{

state = false;

mount = false;

audios = NULL;

m = NULL;

TotalSongs = 0;

}

void setMount(bool n)

{

mount = n;

}

void Power(bool n)

{

state = n;

}

void setDevice(string s, string directory)

{

if ((s == "USB") || (s == "usb"))

m = new USB(directory);

else if ((s == "CD") || (s == "cd"))

m = new CD(directory);

}

void getAudio()

{

audios = m->getAudios(TotalSongs);

}

void Play()

{

for (int i = 0; i < TotalSongs; i++)

PlaySound((s2ws(audios[i]).c\_str()), NULL, SND\_SYNC);

}

};

class InCarAudioPlayer2 {

bool Playlist;

public:

InCarAudioPlayer2()

{

Playlist = false;

}

};

class InCarAudioPlayer2 :public InCar

{

bool playlist\_loaded;

public:

InCarAudioPlayer2()

{

playlist\_loaded = false;

}

void SavePlayList(string Name)

{

char\* n = &Name[0];

bool c = \_mkdir(n);

if (c)

cout << "Playlist Created\n";

}

void AddAudio(string Playlist\_Name, string Audio\_Name)

{

string directory = "d:\\" + Playlist\_Name + "\\" + Audio\_Name;

string src = m->getDirectory();

CopyFile(s2ws(src).c\_str(), s2ws(directory).c\_str(), true);

}

void Delete(string Name, string Audio)

{

string directory = "d:\\" + Name + "\\" + Audio;

char\* ch = &directory[0];

remove(ch);

}

void Shuffle()

{

unsigned seed = 0;

int n = TotalSongs;

// Shuffling our array

shuffle(audios, audios + n,

default\_random\_engine(seed));

}

};

int main()

{

}