#include <iostream>

#include <string>

#include <vector>

#include <cstdlib>

#include <algorithm>

#include <fstream>

#include <conio.h>

using namespace std;

class AudioPlayer { //BASE CLASS AUDIO PLAYER

protected:

string manufacturerName;

string modelNumber;

string storageMedia; ///the data members of base class

string currentSong;

bool powerStatus;

public:

AudioPlayer() {

manufacturerName = "";

modelNumber = "";

storageMedia = ""; //constructor of base class

currentSong = "";

powerStatus = false;

}

///SETTER FUNCTION FOR MODEL AND MANUFACTOR

void setManufacturerName() {

cout << "Enter the manufacturer name: ";

cin >> manufacturerName;

}

void setModelNumber() {

cout << "Enter the model number: ";

cin >> modelNumber;

}

////function that check the input device use for storge media

void setStorageMedia() {

cout << "Enter the storage media (CD/USB): ";

cin >> storageMedia;

if (storageMedia != "CD" && storageMedia != "USB") {

cout << "Invalid storage media. Only CD and USB are supported." << endl;

int x;

cout << "if you want to close media player enter 0 otherwise 1 = ";

cin >> x;

if (x == 0)

{

powerOff();

}

if (x == 1)

{

setStorageMedia();

}

}

}

/// FUN THAT play THE RESENT ENTER THE SONG

void setCurrentSong() {

cout << "Enter the current song that you want to play : ";

cin >> currentSong;

}

///fun for power on the media player

void powerOn() {

powerStatus = true;

}

//fun for power off the media player

// Also move backward the media player

//Also call the presiststate

void powerOff() {

cout << "the car kit poweroff"<<endl;

powerStatus = false;

int x;

cout << "if you want to on media player enter 1 other wise 0 = ";

cin >> x;

if (x == 1)

{

setStorageMedia();

}

persistState();

}

// persistState fun that write the data in the file

void persistState() {

ofstream outfile;

outfile.open("playerstate.txt");

outfile << manufacturerName << endl;

outfile << modelNumber << endl;

outfile << storageMedia << endl;

outfile << currentSong << endl;

outfile.close();

}

// restoreState FUN THAT READ THE DATA FROM FILE

void restoreState() {

string line;

ifstream infile;

infile.open("playerstate.txt");

getline(infile, line);

manufacturerName = line;

getline(infile, line);

modelNumber = line;

getline(infile, line);

storageMedia = line;

infile.close();

}

//FUN THAT DISPLAY THE CURRENT SONG

void displayCurrentSong() {

cout << "Playing: " << currentSong << endl;

}

//MOUNTED AND UNMOUNTED FUNCTION

void mountMedia() {

if (storageMedia == "CD" || storageMedia == "USB") {

cout << "Media mounted." << endl;

}

else {

cout << "Invalid storage media. Only CD and USB are supported." << endl;

}

}

void unmountMedia() {

currentSong = "";

cout << "Media unmounted." << endl;

}

};

///////// "DERIVED CLASS THAT IS PUBLIC INHERITED FROM THE BASE CLASS" /////////

class AudioPlayerV2 : public AudioPlayer {

private:

vector<string> playlist; // VECTOR HAS TYPE STRING THAT IS DATA MEMBER OF DERIVED CLASS USE TO STORE DATA HAS DYNAMIC SIZE

public:

// shufflePlaylist FUN IS USED TPO PLAY THE ALL PLAYLIST FROM BEGIN TO END

void shufflePlaylist() {

random\_shuffle(playlist.begin(), playlist.end());

}

// savePlaylist FUN THAT HAS TASK TO WRITE THE DATA IN FILE FROM "VECTOR"

void savePlaylist(string filename) {

ofstream outfile;

outfile.open(filename);

for (string song : playlist) {

outfile << song << endl;

}

outfile.close();

}

// loadPlaylist FUN IS USED TO GRT DATA FROM FILE AND ADD IN THE VECTOR THROUGH PUSH BACK BUTTON

void loadPlaylist(string filename) {

string line;

ifstream infile;

infile.open(filename);

while (getline(infile, line)) {

playlist.push\_back(line);

}

infile.close();

}

// ADD AUDIO FUN IS USED TO ADD NEW SONG IN THE PLAYLIST

bool addAudioFile() {

string song;

string choice;

cout << "Do you want to add song? (y/n): ";

cin >> choice;

while (choice == "y") {

cout << "Enter the song to add: ";

cin >> song;

playlist.push\_back(song);

cout << "do you want add more song (y/n): ";

cin >> choice;

}

return 0;

}

// DELETE AUDIO FUN IS USED TO DELETE ANY SONG FROM THE PLAYLIST

bool deleteAudioFile() {

string song;

string choice;

cout << "Do you want to delete more song? (y/n): ";

cin >> choice;

while (choice == "y") {

cout << "Enter the song to delete: ";

cin >> song;

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

if (playlist[i] == song) { //FOR LOOP IS USED TO CHECK THE ALL VECTOR TO SEARCH THE SONG

playlist.erase(playlist.begin() + i); //IF LOOP USED TO DELETE THE SONG FROM THAT i POSITION

break;

}

}

cout << "Do you want to delete more song? (y/n): ";

cin >> choice;

}

return 0;

}

// SEARCH AUDIO FUN IS USED TO SEARCH SONG IN THE PLAYLIST

//IF NOT PRESENT THAT GIVE SIGN

//IF FOUND THAT ALSO ASK IF YOU WANT TO PLAY THAT SEARCH SONG

bool searchAudioFile() {

string song;

string choice;

do {

cout << "Enter the song to search: ";

cin >> song;

bool found = false;

for (string s : playlist) {

if (s == song) {

found = true;

break;

}

}

if (found) {

cout << song << " is in the playlist." << endl;

cout << "Do you want to play the song? (y/n): ";

string playchoice;

cin >> playchoice;

if (playchoice == "y") {

currentSong = song;

setCurrentSong();

displayCurrentSong();

}

}

else {

cout << song << " is not in the playlist." << endl;

}

cout << "Do you want to search moresong? (y/n): ";

cin >> choice;

} while (choice == "y");

return 0;

}

};

int main() {

AudioPlayerV2 player; // DERIVED CLASS OBJECT

int x;

cout << "if you want to on the media player enter the 1 otherwise 0 = ";

cin >> x; // CONDITION FOR LOOP OF ALL MAIN FUNCTIONALITY

while (x)

{

cout << " .... WELLCOME TO MEDIA PLAYER .... " << endl << endl;

player.powerOn(); //POWERON FUNCTION CALL

player.setManufacturerName(); //SETTER FUN CALL

player.setModelNumber();

player.setStorageMedia();

player.mountMedia(); // CALL FUNCTION FOR MOUNTED MEDIA PLAYER

player.setCurrentSong(); // CALL FUNCTION TO SET CURRENT SONG IN MEDIA PLAYER

player.displayCurrentSong();

player.unmountMedia(); // CALL FUNCTION FOR UNMOUNTED MEDIA PLAYER

player.restoreState(); // CALL FUNCTION FOR READ THE DATA FROM FILE

player.addAudioFile(); // CALL FUNCTION FOR ADD THE DATA IN VECTOR ARRY

player.savePlaylist("playlist.txt"); // CALL FUNCTION FOR WRITE THE DATA IN FILE FROM VECTOR ARRY

player.loadPlaylist("playlist.txt"); // CALL FUNCTION FOR WRITE THE DATA FROM FILE FROM IN VECTOR ARRY

player.shufflePlaylist(); //CALL FUN THAT SHUFFLE THE PLAYLIST FROM BEGIN TO END

player.deleteAudioFile(); //FUN FOR DELETE THE SPECIFIC SAVE DATA

player.searchAudioFile(); //CALL FUN TO SEARCH THE PARTICULAR SONG

// CON FOR WHILE LOOP

cout << "if you want to off media player enter 0 otherwise 1 = ";

cin >> x;

//CLEAR SCREEN COMMMAND

system("CLS");

}

player.powerOff(); // CALL POWEROFF FUN IN THE END

return 0;

}