Erick Cabrera

Prof. Huang

Final Project

April 30, 2017

**Source Code:**

**Main class**

/\*

\* main.cpp

\* Created on: Apr 25, 2017

\* Author: Erick Cabrera

\*/

#include <iostream>

#include <string.h>

#include "TV\_Set.h"

#include "ChannelArrayList.h"

using namespace std;

TV\_Set \*tv;

void printMenu(){

cout << "1 - Turn TV off/on"<<endl;

cout << "2 - go to a different channel"<<endl;

cout << "3 - go up a channel"<<endl;

cout << "4 - go down a channel"<<endl;

cout << "5 - go to a different sound volume level"<<endl;

cout << "6 - go up a sound volume level"<<endl;

cout << "7 - go down a sound volume level"<<endl;

cout << "8 - add a new channel"<<endl;

cout << "9 - remove a channel"<<endl;

cout << "10 - display all channels"<<endl;

cout << "11 - what am I watching right now?"<<endl;

cout << "12 - quit"<<endl;

}

void doOption(int choice){

switch(choice){

case 1:

tv->togglePower();

if(tv->isTvOn())

cout<<"TV is now on!\n";

else

cout<<"TV is now off!\n";

break;

case 2:

if(tv->isTvOn()){

int channel;

cout<<"Please enter a channel\n";

cin>>channel;

tv->changeChannel('0',channel);

}else

cout<<"TV is off, cannot function in this state!\n";

break;

case 3:

if(tv->isTvOn()){

tv->changeChannel('u',0);

}else

cout<<"TV is off, cannot function in this state!\n";

break;

case 4:

if(tv->isTvOn()){

tv->changeChannel('d',0);

}else

cout<<"TV is off, cannot function in this state!\n";

break;

case 5:

if(tv->isTvOn()){

int volume;

cout<<"Please enter a volume\n";

cin>>volume;

tv->changeVolume('0',volume);

}else

cout<<"TV is off, cannot function in this state!\n";

break;

case 6:

if(tv->isTvOn()){

tv->changeVolume('u',0);

}else

cout<<"TV is off, cannot function in this state!\n";

break;

case 7:

if(tv->isTvOn()){

tv->changeVolume('d',0);

}else

cout<<"TV is off, cannot function in this state!\n";

break;

case 8:

if(tv->isTvOn()){

int channel;

cout<<"Please enter a channel number and programming title\n";

cin>>channel;

string title;

getline(cin,title);

char\* cp = new char[title.length()+1];

strcpy(cp,title.c\_str());

tv->addChannel(channel,cp);

}else

cout<<"TV is off, cannot function in this state!\n";

break;

case 9:

if(tv->isTvOn()){

int channel;

cout<<"Please enter a channel to remove\n";

cin>>channel;

tv->removeChannel(channel);

}else

cout<<"TV is off, cannot function in this state!\n";

break;

case 10:

if(tv->isTvOn()){

tv->displayAll();

}else

cout<<"TV is off, cannot function in this state!\n";

break;

case 11:

if(tv->isTvOn()){

tv->tvStatus();

}else

cout<<"TV is off, cannot function in this state!\n";

break;

}

}

int main(int argc, char\*\* argv){

int maxVolume = 0;

if(argc == 2){

maxVolume = atoi(argv[1]);

}

if(maxVolume <= 0)

maxVolume = 25;

tv = new TV\_Set(maxVolume);

while(1){

printMenu();

int choice;

cin >> choice;

if(choice >12 || choice < 1)

cout << "Not a valid option\n";

if(choice==12)

break;

else

doOption(choice);

}

cout <<"Exiting!\n";

return 0;

}

**TV\_Set class**

/\*

\* TV\_Set.cpp

\* Created on: Apr 25, 2017

\* Author: Erick Cabrera

\*/

#include "TV\_Set.h"

#include "ChannelArrayList.h"

#include <iostream>

using namespace std;

TV\_Set::TV\_Set(int maxVolume){

isOn = true;

this->maxVolume = maxVolume;

currentVolume = maxVolume/2;

channels = new ChannelArrayList();

currentChannel = 0;

}

TV\_Set::~TV\_Set(){

delete(channels);

}

void TV\_Set::addChannel(int channelNumber, char\* programmingTitle){

ChannelArrayList::Channel \*newChannel = new ChannelArrayList::Channel;

newChannel->channelNumber = channelNumber;

newChannel->programmingTitle = programmingTitle;

channels->add(newChannel,true);

if(currentChannel == 0)

currentChannel = newChannel;

channels->findNewIndex(currentChannel);

}

void TV\_Set::changeChannel(char direction, int specificChannel){

if(direction == 'u'){

currentChannel = channels->changeChannel("up");

}

else if(direction == 'd'){

currentChannel = channels->changeChannel("down");

}

else {

currentChannel = channels->getChannel(specificChannel);

}

channelStatus();

}

void TV\_Set::changeVolume(char direction, int speficiVolume){

if(direction == 'u'){

currentVolume++;

if(currentVolume > maxVolume)

currentVolume = maxVolume;

volumeStatus();

return;

}

if(direction == 'd'){

currentVolume --;

if(currentVolume < 0)

currentVolume = 0;

volumeStatus();

return;

}

if(direction == 'm'){

currentVolume = 0;

volumeStatus();

return;

}

if(speficiVolume >= maxVolume)

currentVolume = maxVolume;

else if(speficiVolume <= 0)

currentVolume = 0;

else

currentVolume = speficiVolume;

volumeStatus();

}

bool TV\_Set::isTvOn(){

return isOn;

}

void TV\_Set::removeChannel(int specificChannel){

if(specificChannel == currentChannel->channelNumber){

changeChannel('d',0);

}

channels->remove(specificChannel);

}

void TV\_Set::displayAll(){

channels->toString();

}

void TV\_Set::volumeStatus(){

cout << "Sound volume @" << currentVolume<<endl;

}

void TV\_Set::channelStatus(){

if(currentChannel)

cout << "You are currently watching "<<channels->toString(currentChannel)<<endl;

else

cout << "OOPS! this channel is not available.\n";

}

void TV\_Set::togglePower(){

isOn = !isOn;

}

void TV\_Set::printStatus(){

if(currentChannel)

cout << "You are currently watching "<<channels->toString(currentChannel) << ", sound volume @" << currentVolume<<endl;

else

cout << "OOPS! this channel is not available.\n";

}

void TV\_Set::tvStatus(){

printStatus();

}

**ChannelArrayList class**

/\*

\* ChannelArrayList.cpp

\* Created on: Apr 24, 2017

\* Author: Erick Cabrera

\*/

#include "ChannelArrayList.h"

#include <cstdlib>

#include <iostream>

#include <string.h>

#include <stdio.h>

using namespace std;

ChannelArrayList::ChannelArrayList(){

size = 0;

currentChannel = 0;

lastAddedIndex = -1;

}

ChannelArrayList::~ChannelArrayList(){

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

delete(channels[i]);

delete [] channels;

}

bool ChannelArrayList::add(struct Channel \*channel,bool forceAdd){

if(size >= MAX\_CHANNELS && !forceAdd)

return false;

//check for dupes

if(size>=1){

if(int index = checkForDupes(channel)>=0){

channels[index] = channel;

return true;

}

}

//check for null spaces

if(lastAddedIndex>=MAX\_CHANNELS-1){

bool reset = true;

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

if(channels[i]==NULL){

lastAddedIndex = i-1;

reset = false;

}

}

if(reset){

lastAddedIndex = -1;

}

}

lastAddedIndex++;

cout<<lastAddedIndex<<endl;

if(size < MAX\_CHANNELS)

channels[lastAddedIndex] = (ChannelArrayList::Channel\*)malloc(sizeof(ChannelArrayList::Channel));

channels[lastAddedIndex] = channel;

if(size<MAX\_CHANNELS)

size++;

this->sort();

return true;

}

ChannelArrayList::Channel ChannelArrayList::remove(int channelNumber){

struct Channel removedChannel;

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

if(channels[i] != 0 && channels[i]->channelNumber == channelNumber){

int index = i;

removedChannel = \*channels[i];

delete(channels[index]);

for(int j = i+1; j<size; j++){

channels[index] = channels[j];

index++;

}

size--;

if(i!=size)

delete(channels[size]);

this->sort();

return removedChannel;

}

}

return removedChannel;

}

ChannelArrayList::Channel\* ChannelArrayList::getChannel(int channelNumber){

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

if(channels[i] != 0 && channels[i]->channelNumber == channelNumber){

struct Channel \*removedChannel = channels[i];

currentChannel = i;

return removedChannel;

}

}

return NULL;

}

ChannelArrayList::Channel\* ChannelArrayList::changeChannel(char\* direction){

if(strcmp(direction,"up")==0){

if(currentChannel >= size-1)

currentChannel = 0;

else

currentChannel++;

}else{

if(currentChannel <= 0)

currentChannel = size-1;

else

currentChannel--;

}

struct Channel \*changedChannel = channels[currentChannel];

return changedChannel;

}

std::string ChannelArrayList::toString(struct Channel \*channel){

char num[10];

string toString("Channel ");

snprintf(num, 10, "%d", channel->channelNumber);

toString.append(num);

toString.append(" - ");

toString.append(channel->programmingTitle);

return toString;

}

void ChannelArrayList::toString(){

if(size == 0)

return;

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

cout << toString(channels[i]) << endl;

}

}

void ChannelArrayList::sort (){

int length = size;

ChannelArrayList::Channel \*temp;

int j;

for (int i = 0; i < length; i++){

j = i;

while (j > 0 && channels[j]->channelNumber < channels[j-1]->channelNumber){

temp = channels[j];

channels[j] = channels[j-1];

channels[j-1] = temp;

j--;

}

}

}

void ChannelArrayList::findNewIndex(struct Channel \*channel){

if(!channel){

currentChannel = 0;

return;

}

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

if(channels[i]->channelNumber == channel->channelNumber){

currentChannel = i;

return;

}

}

}

int ChannelArrayList::checkForDupes(struct Channel \*channel){

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

if(channels[i]->channelNumber == channel->channelNumber)

return i;

}

return -1;

}

**TV\_Set header class**

/\*

\* TV\_Set.hpp

\* Created on: Apr 25, 2017

\* Author: Erick Cabrera

\*/

#ifndef TV\_SET\_H\_

#define TV\_SET\_H\_

#include "ChannelArrayList.h"

class TV\_Set{

public:

TV\_Set(int maxVolume);

~TV\_Set();

void addChannel(int channelNumber, char\* programmingTitle);

void changeChannel(char direction, int specificChannel);

void changeVolume(char direction, int speficiVolume);

bool isTvOn();

void volumeStatus();

void channelStatus();

void tvStatus();

void togglePower();

void removeChannel(int specificChannel);

void displayAll();

private:

ChannelArrayList::Channel \*currentChannel;

int maxVolume;

bool isOn;

int currentVolume;

ChannelArrayList \*channels;

void printStatus();

};

#endif /\* TV\_SET\_HPP\_ \*/

**ChannelArrayList header class**

/\*

\* ChannelArrayList.hpp

\* Created on: Apr 24, 2017

\* Author: Erick Cabrera

\*/

#ifndef CHANNELARRAYLIST\_H

#define CHANNELARRAYLIST\_H

#include <cstdlib>

#include <string>

using namespace std;

#define MAX\_CHANNELS 10

class ChannelArrayList{

public:

struct Channel{

int channelNumber;

char\* programmingTitle;

};

ChannelArrayList();

~ChannelArrayList();

bool add(struct Channel \*channel, bool forceAdd = false);

struct Channel remove(int channelNumber);

struct Channel\* getChannel(int channelNumber);

struct Channel\* changeChannel(char\* direction);

std::string toString(struct Channel \*channel);

void toString();

void findNewIndex(struct Channel \*channel);

private:

int size;

int currentChannel;

int lastAddedIndex;

struct Channel \*channels[MAX\_CHANNELS];

void sort();

int checkForDupes(struct Channel \*channel);

};

#endif /\* ChannelArrayList\_HPP\_ \*/

**Output**



















